Observational Studies On Onboarding.

Onboarding Event

Typical Observation study at Berlin Germany, OpenStack Upstream Institute (OUI).

Mentors
: (M1, M2, …, M12)

Participants
: (P1, P2, …, P72), 17 females, 23 neutrals, and 32 males seated on 12 tables forming 12 groups

Outline

Theoretical Knowledge (Day-01)

- Introduction
- Accounts creation and setup
- Setting up the Development Environment
- How OpenStack is Made
- OpenStack Events

Practical Knowledge (Day-02)

- Workf-low and Tools for participation
- Code Dive Deep
    - Technical Activity

DAY-01 Theoretical Activities

SESSION-START 12:00 PM, November 09-10, 2018.

PRESENTATION of mentors and the training program

- Introduction

- Mentors :: The coordinators (two lead mentors) welcomed and thank everyone for coming, then present the mentors to the general audience. These mentors constitute project team leads (PTLs) of different OpenStack projects and cross-project teams and other resources within the OpenStack Ecosystem.

- Setting the Working Environment

- Working Environment :: Mentors ensure that the working environment of each participant is properly set up as required by the instructions that were sent out one month prior to the onboarding event.

This working environment constitutes a laptop that supports virtual environments such as VirtualBox with Ubuntu image pre-installed, a copy of OpenStack development environment aka Devstack on Sandbox, issue trackers (both launchpad and storyboard), code review environment (Gerrit), git. The OpenStack Sandbox environment (repository) provides virtual servers to test OpenStack projects/functionalities in an isolated environment. Therefore, contributors can test specific features without worrying about the production environment (repository).

M1 said "Make sure the following are install and running:-

- Install Virtualbox on your host machine with the Ubuntu image
- Install DevStack on Sandbox Environment

- Install Git, Gerrit, Python3, Editor (Vim, Sublime, SPE, or any other open-source tool)

Run the test script in the Devstack folder to make sure your local environment is properly configured and up to date. In case you run into trouble, call any mentor to help you."

Then, mentors guided the participants in joining the OpenStack foundation and creating their accounts with IRC Freenode, OpenStack community mailing list after reading and signing all the necessary agreements.

Each mentor takes their rounds and introduced themselves, their area of expertise, how many projects they are affiliated to and how long they have been contributing to the OpenStack ecosystem.

Day one is typically designed to give a solid foundation of OpenStack and how to set up an individual working station. Moreover, the mentors also presented how bugs are reported to the community, (how to report a bug), triage, and assigned a bug. Furthermore, M1 and M2 presented an overview and objectives of the OpenStack Upstream Institute (OUI) Program.

Understand the OpenStack release cycle to the level of being able to synchronize and integrate it with your product's roadmap Get to know the technical tools Understand the OpenStack contribution workflow and social norms Know where to find information, where and how to get help if needed Be able to identify and start a task (bug fix, feature design and implementation, Working Group activity and so forth) Work-flow and Tools for Participation

M1

> "The OpenStack Upstream Institute was designed by the OpenStack

foundation to share knowledge about the different ways to contribute to OpenStack. The program was built with the principle of open collaboration in mind and was designed to teach attendees how to find information, as well as how to navigate the intricacies of the technical tools for each project. The training program to share knowledge about the different ways of contributing to OpenStack like providing new features, writing documentation, and participating in working groups."

M2

> "OUI is about to help new contributors to join the OpenStack community by providing on-site training for newcomers and place for trainers and mentors to work together."

M1

> "We have seen some PTLs coming back to OUI training for some kings of reality checks."

Moreover, M2 emphasized on IRC and the mailing list as the main communication Medium. Also, "We strongly recommend the constant consultation of the online documentation as we ourselves are constantly referencing them throughout this training event. Read, read and Read your documentations!"

- How OpenStack is Made

TASKING : How mentors coordinated activities

The Mentors took their turn and presented the different projects working together to form the complex ecosystem. Also, each mentor states how many project(s) that they are involved with and explain how the projects fit together under one umbrella called OpenStack. Moreover, **M1** The ecosystem lead at OpenStack gave use cases to participants explaining how individual project goals differ from cross-project goals, also, the cross-projects common objectives build a community into an ecosystem. **M1** said: " Both an individual project and an ecosystem do have their respective communities, that is the people who are making things to happen, not only developers but contributors in all forms. However, an individual project follows one particular design paradigm and common work-flow, which all the community members are accustomed to. No matter how big an individual project may be in terms of the size, for example, the Linux kernel, every member in that community still follows one work culture and common objectives. However, this is not the case for an ecosystem such as OpenStack, where all the individual projects forming OpenStack have their own individual plan-of-action and work culture, but need some coordination to project one common product at the end of each release cycle. Besides, in an ecosystem, the design paradigm is different and depends on domain knowledge. In addition, In an ecosystem, cross-project collaboration is the force that builds a community into an ecosystem but such is not the case with an individual project; they don't have cross-projects. There are several other differences that exist between an individual project and an ecosystem, some we will revisit later."

M1 added that: "If we remember the early days of the internet, it was the LAMP (Linux, Apache, MySQL, PHP) stack that enabled the rapid growth of the Web. In this era of cloud computing, OpenStack is the 'LAMP stack' of the cloud. The same way the Linux kernel is different from Apache server, and MySQL DB, and PHP, so too is how the different projects within the OpenStack ecosystem differ from one another. Yes, all use Python but that is it."

Assigning tasks to participants

During each presentation, the main mentor responsible for that presentation tasked the participants on micro-tasks. For example, a mentor **M3** asked the participants to go on-line and search the release cycle of OpenStack and how many releases are there in total. #DURATION 2 min

In addition, **M6** asked participants how many core projects exist at OpenStack and how who are the project team leads (PTL) of Nova, Swifts, Cinder, Neutron, and Manila?

The first participant who submitted the right answer on the IRC channel was rewarded with a sticker, and the same goes for all the other mentors as they did their presentations on the structure and functioning of OpenStack cross-project teams and the ecosystem in general.

After the set duration, if none of the participants found the correct answer, the mentor will tell them what the correct answer was.

For example, M3, M5, M7, M6, and M9 all assigned specific micro tasks (tasks that usually last two min) to participants to carry out. The hands-on is programmed for Day-02 with much intense tasks and workload.

Mentees configuration

new contributors were organized in a cluster of six per table and there were 12 tables in a spacious room that can support about 150 attendees in general.

LEARNING-METHOD How new contributors are adapting within the ecosystem.

problem-based learning helping students to find solutions for the problem themselves.

OB1: Why was the mentoring program introduced? M1: "We wanted to lower the entry barriers that new contributors were having – speaking from experience, we discussed among ourselves and put down several points that can be beneficial for new contributors."

FEEDBACK and Testimonies from mentors who were mentees previously.

M7

said "After the 2-Days onboarding event, participants can sign up for a longer-term mentoring program to further strengthen their skills and become more productive and successful in the community. — That's the way to transform learners into practitioners"

M4

Getting involved with the OpenStack community can be a daunting task. Where does a newcomer begin? While the OpenStack community does provide a rich set of resources to newcomers, like the OpenStack Upstream Institute, getting the courage to take the leap into the world of open infrastructure can prove difficult.

M9

gave her perspective as a fresh university graduate when she first joint OpenStack community as a new contributor, she said "mentoring did not only plays a strong role in the workplace where I am now working on four OpenStack projects but also within the OpenStack community at large, where mentoring has played a major role in my journeys towards making my first contributions to OpenStack."

TESTIMONY1 (~~M9 testimony~~)

So far, we've given you some perspective on our experiences with OpenStack to date we're going to get to the juicy parts and why we feel mentoring upstream is important which is I assume why you're here.

Ultimately, in my opinion, I feel that mentoring individual contributors upstream adds value both to the open infrastructure community and the individual projects that make up this diverse infrastructure ecosystem we've come to love and sometimes love to hate.

This value comes in many forms one of those I believe is growing the number of individual contributors helps drive these projects forward through more diverse reviews, contributions, and viewpoints. By expanding that diversity we're able to expand ultimately the diversity of opinions for the open infrastructure project as a whole with the goal of as soon as we start at least, in my opinion, is we grow the diverse opinions that we have it should hopefully start to attract additional individual contributors as our solutions expand to cover more use cases.

Upstream mentoring also helps us grow the expertise the community needs ultimately the technical expertise required to work with the technologies that make up the projects under the open infrastructure umbrella it can be pretty vast that technical expertise then helps us drive the projects themselves for deployment projects especially such as OpenStack helm and Kola kubernetes and some of the others that I've worked within the past also require a working understanding of technologies that extend beyond just you know OpenStack.

To successfully drive these projects forwards toward their medium and long term goals it only makes sense to invest our time and help our individual contributors grow the technical skills required to do so active mentoring also helps lower the barrier to entry for individuals looking to contribute to open infrastructure projects.

Speaking from experience finding projects and work that align with personal or work interest ~~improve~~ involves difficult if you have little to no experience with what the community has to offer the first question I always ask when someone expresses interest in working with OpenStack helm is what interest and excites you I asked this question because I found that there's if there's no interest or excitement for the work you're looking to do it often leads to frustration when you start running into problems in difficult situations and you just walk away from the project entirely.

In addition to helping find work that aligns with an active mentor and can help ensure new contributors are visible to the larger project teams and community and once again my personal experience has been that once I felt included and comfortable with the Kola project team, in particular, my engagement and contributions accelerated rapidly from that point.

TESTIMONY2 (~~FP1 testimony~~)

So, as a new developer fresh out of college coming into any new team can be very intimidating. Everyone around the kind of knows so much more than you and you feel that you're an imposter with so much to learn there's just no way you can learn everything that they know and you're under the impression that that knowledge is somehow inherent to them and they just get it and you're just never will.

Now, magnify that and add people you don't interact with face to face and you have what it's like to enter a community of OpenStack or any open source project. It's very intimidating you don't know who to turn to or what questions to even ask.

So, contributing to an upstream project is so much more than just being added to a new team, there are now people all over the world that you have to deal with. It's a lot like having another person act as a mentor is like having an interpreter.

It provides someone you can ask questions without worrying if they're going to think you just have no idea what you are doing. A mentor provides a way to get involved and helps remove so much of the anxiety that often comes with doing so.

It's really just having a kind of cushion that you get to bounce some ideas off of to check your understanding but not only that it's it changes the entire experience of from a scary one to one that definitely seems manageable there are a lot of important conversations that happen around a project from weekly meetings to attending summits a lot goes on around the project that you can easily miss if you don't know it's happening, having a mentor helped me be active in meetings push me to do things like submit and then give my first commit.

So, there are all these things that happen you don't necessarily know that they're going on when you find out they're going on you don't think that you have any reason to be involved because obviously you know nothing so what could you contribute but it's really important to be a part of that conversation having a mentor to push you to do that is very helpful.

TESTIMONY3 (~~M7 testimony~~)

So, I'll talk a little bit about my experience when it comes to mentoring upstream after reflecting on my experience with previously you know the OpenStack community and now the open infrastructure community over the past four years the experiences I've had during that time was the primary driver for wanting to encourage you on your own personal journey. The following points are the ones that really stick out in my mind thinking back over what's happened over the last four years or so. The biggest one is time constraints, most of us mentors have day jobs, families and hobbies that may not include hacking away at the keyboard, because of that finding time for active mentoring can be a challenge. I found that carving out time for that act of mentoring though has worked the best for me as I'm historically terrible at saying time isn't an issue so I'll just do this mentoring thing live. But, if you want to be a successful mentor though or a mentee even, my advice is simple as with testing and production don't do it.

I've also found successful mentoring requires active commitment both from the mentor and mentee. It's very easy to say yeah sure I'd love to help you understand all this cool stuff. It's not really that hard but it can be difficult to deliver on that promise.

People learn in different ways at different speeds which means a commitment to active mentoring requires more than a handful of quick IRC or Google hangout Chats when our time constraints increase and we start wishing we had 25 hours on a given day to get everything done that we need to.

It's often tempting to ease off things like mentoring as a way to get some of that time back. For those who have mentored me in the past, I'm really grateful that that wasn't their approach. As a mentee, I've been guilty of thinking also this person is super busy doing super cool stuff and because they're doing all this super cool stuff and they're super busy I don't want to get in the way of that so I'm just going to leave them alone. Commitment has to be two-way traffic for mentoring to succeed.

Also, as a mentor, I've noticed it's easy to overlook the differences and the technical proficiencies between myself and someone I'm working to mentor. It wasn't uncommon for me in the past to take the years of experience I have for granted and just assume everyone at least knew half or more of what I do. Remember the question earlier that I said I'd like to ask each new individual contributor who reaches out to me asking about their interest and experience provides me an entry point for digging deeper into what experience a potential mentee has.

I found that being proactive about that and managing those expectations has worked the best for having successful mentor-mentee relationships.

FP1's Company perspective: So now that we've talked about mentoring upstream, let's also talk about why it's equally important to have a mentor in your company. Mentoring in the workplace goes just beyond helping with delivering day-to-day duties. Active workplace mentoring ultimately helps both mentees and mentors alike to achieve their personal career goals.

Active workplace mentoring helps mentees attain mature technical skills required to grow in their workplace, mentoring helps manage immature skill sets required to grow into a senior engineering role in the future. So, the maturation of those technical skills may also help alleviate impostor syndrome as most of us are likely familiar with.

Mentoring also helps senior engineers grow their leadership and delegation skills which may help them grow into leadership engineering or even management roles in the future.

Mentoring is also a sound business investment. Teams and enterprises cannot afford to lose their top engineering talent as the needs of the business evolve, especially in industries where disruptive technologies result in an extremely competitive pool of talent. Growing talent through mentoring is a medium to long term investment.

However, the return of that investment can be very high. Investment in mentoring is key to staying competitive and keeping employees happy so in the long run yes mentoring can be an investment on behalf of you know the company but it pays out dividends later on. So, this is my mentoring experience in the workplace, it's been very unique.

M7 was my mentor during my last year of college and I have been very fortunate working with him and to continue being his mentee. In college, I was fully prepared to accept the job that I was interning at on a project that really just didn't excite me and I sort of went to 95 and said okay that well that's it I'll never be excited about what I'm gonna do.

I didn't think I was someone who was capable of contributing to something bigger, but having a mentor I began to figure out almost everyone felt that same way. The difference was putting in the effort to put me out there and learn something new. Without that push, I probably wouldn't have left my little bubble and would have thought I was not capable of anything else.

I'm still nervous about submitting a patch that's for review but I'm a lot less nervous than I was previously and I understand now that people on my project aren't sitting there silently judging me for not knowing all the answers but are actually rooting for my success and happy to help what I have questions now I'm able to actually go to other people when I have questions if **M7** is busy and I'm happy to get reviews where people point things out that I can improve on I've learned so much through mentoring and I'm incredibly happy to continue to learn I have now this group of people that I can draw knowledge from more than just what I started with that and I think the biggest benefit of mentoring is the connections I have now made that I can go to.

So, there are a lot of times where issues arise or defects occur in a workplace and shadowing to fix them is probably the best way to learn how to deal with them in the future learning where to go to look for problems what things to watch for and what they mean how to interpret them is a very long process and shadowing someone else who knows how to do it is a great way to figure it all out.

Don't assume an answer to a question should be obvious and it's happening multiple times now where I have encountered an issue and get in my own head and I should be able to fix this without asking any questions only to exhaust every avenue that I now bring up the issue and find out something else was going on that I didn't know about causing my issue.

Therefore, I have to not be afraid to speak up this is why it's really important to speak up and communicate if my mentor doesn't know I'm struggling with an issue he can't help me so making sure I say something when it happens or even if I figured it out later but say this was an issue for me I figured it out that sort of communication is very vital and then there happens only occasionally.

But, there are times where sometimes my mentor is doing something that I sit there and go. I think it's supposed to go this way so if I speak up at those points instead of sitting there silently it's actually a good thing because I'm learning and we don't have to sit on the same issue forever.

So, learning the issue, seeing someone else make the same mistake and going away I know that that's a sign of progress and something I should be happy with not internalizing.

TESTIMONY4 (~~M7 testimony~~)

So, I'm going to share a little bit about some experience of mine with mentoring in the workplace. My experience at XYZ company. When I started there I didn't have a mentor to really turn to work on the community team that I mentioned where my work was focused solely on upstream projects that I found interesting and thought might bring value back to the company.

Down the road I was fairly insulated from the internal development and deployment efforts of the business often, I would find myself running into problems that I was too timid or afraid to ask for help with is I didn't want to go and interrupt the guys who had been working 60-70 hours a week trying to solve issues heading up to up to deadlines.

In my mind, they're working to solve the real problems and I'm sitting here you know playing upstream and having a good time fast forward to now though I make it a point to actively help both junior members of staff and our organization and also my peers who are working to understand the technologies we rely on for what our business is trying to achieve.

Now, I can sympathize with my coworkers and peers because I remember how I felt when I first started and I was this bear in this picture just sitting around thinking you know I don't have a mentor here I'm just gonna sit and wait until someone says something because I'm sure they can look at me and tell that I'm stuck.

It wasn't the case, it was very easy to feel overwhelmed and I try my hardest to see the signs of someone who may need help now but like I did in the past doesn't want to interrupt someone that seems too busy similar to my takeaways from upstream mentoring I've got some downstream takeaways too after reflecting on the past four years some of these are fairly similar to my viewpoints on upstream mentoring.

But, have some slightly different context of course time constraints similar to upstream can affect our ability to mentor our peers in the workplace I feel the time constraints in the workplace can be more restrictive as the time required to be successful often competes with the time required to make the critical needs of the business during the 40 hours a week or so that we're in the office similar to before it often becomes tempting to scale back your involvement with your mentees in the workplace once again my advice is if this can be avoided don't do it.

Bringing a mentee along for the ride when triaging and fixing critical issues in the workplace not only helps them learn the processes required for doing so but also serves as a sanity check for yourself when you inevitably slip up and do something wrong and they pointed out, which as FP1 said it progresses it's great I won't lie there's been a handful of times where that's happened and I'm really glad she was able to catch me on that because I feel like that helps reinforce the concepts that I've got and the things that I've learned and it's a good feeling seeing your mentee catch on as well.

Concerning upstream mentoring, differences and technical proficiency should be taken into account when mentoring in the workplace my approach to addressing this in the workplace is the same as it is with upstream individual contributors I like to find out what interest and excite someone I'm a meant to mentor and find ways to tie that back to the work they're doing if it's not exactly a perfect fit.

I found that personal interests and the technology that we work within our day to day jobs make work feel less like work, which is a good thing.

Also, differences and work preferences can be a challenge. Different people like to work in different ways. I've had successful mentoring arrangements that involve pairing frequently with someone until they felt more confident and comfortable in their abilities.

I've had situations where delegating tasks and activities to a and a more hands-off approach resulted in success where paring had previously failed.

I like to talk with mentees early to determine what works best for them. Putting someone in an extremely uncomfortable situation that might not align with the way they like to work will almost never result in a successful outcome.

For interest in mentees, I highly suggest taking an active approach to finding mentors and coaching in the workplace. Meaningful mentoring arrangements aren't always part of workplace culture.

Successful mentoring relies on active commitment from both parties and I've always been enthusiastic and willing to help anyone who's approached me directly about mentoring because it shows me they take initiative and not only succeeding in their day to day job but they want to actively manage their expectations for their long-term career goals, and finally feedback is critical to determine the effectiveness of a mentoring arrangement.

Not just for the mentee but for the mentor as well the differences and challenges mentioned above will ultimately affect whether you can expect success in a mentoring relationship.

Personally much like software or software development, I prefer to fail fast and regular and get rapid regular feedback from the individuals that I have these relationships in the workplace so I can course-correct and find ways to augment that relationship with my mentee to make it successful because ultimately if in the long run if they're successful everyone in our organization successful it means I get fewer text messages and phone calls at night when things don't work.

So, there's some self-preservation in there as well but mostly it's wanting to see everyone else succeed I am extremely grateful for the experience I've had both upstream and downstream standing here four years later after not just getting involved with OpenStack but graduating college and getting my first job I never thought I'd be standing up here talking about how I can help other people succeed in the workplace and how good of a feeling that is I fully expected to be in a cubicle somewhere just hacking away - keyboard all day and I'm really happy that's not the case.

[Open discussion on Mentoring Q/A] Companies representative, foundation in sum all stakeholders.

Mentoring is super important and you talked about the time trade-off at work like doing that quote-unquote actual work but I'm a firm believer that mentoring as a multiplier is way more valuable than just banging out code. So, thanks for beating the drum. Thanks for taking the mentoring seriously. I really appreciate it. It's awesome.

Thank you yeah I agree 100% thankfully the organization that we work with and that we're a part of firmly believes the same thing and it's just another thing it makes me extremely happy to be where I'm at right now.

without mentoring I would definitely be sitting in a cubicle hacking away at something that oh and know this for sure and not trying to expand my knowledge at all so I feel like I'm actively contributing to something new and that's a great feeling.

Any advice for getting a mentorship program going at a company that doesn't currently have one? that is it's very difficult because it's very much in the culture of the organization that we're part of so it's kind of hard for me to put myself in those shoes and give advice I think the best way to approach it though ultimately is you know pitching it is in the terms of the business it is a solid investment because this was mentioned I mean the time you're putting forth and mentoring people and making sure they're not just able to contribute but they understand that they don't have to be rockstars to add value that really helps get over the hurdles where someone might feel they're inadequate or don't possess the skills or can develop those skills.

but also as FP1 mentioned, working in this industry where sometimes technology can be extremely disruptive and it's very hard to find the right people for the job it for the business it makes it hard to transition to things that add value from a business sense if you're not able to grow the skills in the workplace to handle that if you're not able to find them externally I think that's probably the best way I could answer that.

Yeah, that's a good question if someone else has a better idea or some valid feedback I think it'd be awesome to touch base.

My company does have a formal mentoring program. I do a lot of mentoring for people on my team and one of the elements that we have is that we set up formal goals that are related to the mentoring that change each quarter to align with what we're doing that quarter. Setting formal goals as part of a mentoring program is extremely important and we have recorded lots of successful cases in our company.

One other way to achieve success has been to request a recurring one-on-one with anyone you're mentoring. That could be at whatever frequency they desire. It could be once a week, or once every two weeks, or once a month. It's really just whatever works for both parties and adds value. In the past the mentoring relationships I've ended up establishing with individuals who came into our organization and some of the other people I've met upstream.

Personally, I always kept a curated list of the goals that I wanted to reach. So, for example, starting to get involved with different OpenStack projects and services you know I got involved with OUI and that helped me to get the right sets of projects as I started trying to dive into OpenStack so it was pretty daunting but playing out those granular goals that made sense in a sequential effort I guess really helped. You can't swallow a whale hole right! You gotta cut it up into bits.

I mean setting up goals even just as a mentee yourself and then again communicating those clearly to your mentor I think has been the best way for me to sort of track my progress and stay focused. For example, say I want to submit this many patches upstream and oh I did that and I want to contribute to this or that project at the next release cycle

I was able to do those setting goals like that have been incredibly helpful and have definitely in terms of impostor syndrome. That's the thing that makes me look back and realize oh wow I've actually accomplished a lot so it's really yeah asking people to set goals for themselves I think is helpful.

To the question of how does one set up a mentoring program successfully?

As a former manager, myself and my management colleagues tried to do that exactly the same thing several different ways and tried to you know combinations of like forcing down you're assigned to this person to having people sign up through you know sign-up sheet and anyway I don't think we ever really succeeded in finding something that didn't feel forced and artificial and if anyone has had more success with that I'd be really interested in hearing it but I actually work for the same place as these guys and I think

M7 really captured it. It’s the culture of the place that helps drive the behavior you want you can sort of sell it to leadership as it's an investment and it's it brings you business value and it helps retain talent because it keeps people happy and it increases skills that are hard to find in the marketplace and those are all great ways to basically get permission to allow people to do this but then the other part is encouraging it and rewarding it if you allow people the time you don't make them feel like they are you know taking time away from their actual job but make them understand that it is an encouraging part of their job I think that helps sort of foster that and then if they when they do it if you tell them they did a good job and you know if you're in a management position you're able to reward that kind of behavior that sort of starts that cycle going.

As a follow-up question, do you think it's easier to set up those programs within a given company versus within a given community like is it easier to get management approval for example if you set it up within the same organization how do you think it relates between community to even and company driven mentorships?

If you want my personal opinion, I think it's a lot easier to set up in an open-source community trying to think of the right way to answer this in terms of the company perspective in a large company that has many moving parts and many layers of reporting as a fortune 10 company like AT&T does it could be pretty difficult because you can have you know informal mentoring programs set up at the organization level but you know of course it ties back to what I was saying earlier whereas time constraints get busy if the workforce or the headcount on your team starts to reduce a bit due to the needs of the business usually one of the first things that go is okay well I need you to devote all this time to picking up some of the extra work and focusing on delivering the software we need to deliver and operate the infrastructure we need to operate in this mentoring thing you know we can revisit this later. I think setting it up external to the workplace so in this sense the community –OUI, mostly because the benefit that I have working in the organization that I do the majority of us are working upstream and various projects anyway whether it's OpenStack-helm airship or something else so it's a lot easier to handle it that way and that way we can help mentor our co-workers but also mentor other individual contributors who might want to be involved with that effort as well and then you know since are the people on our team are working upstream anyway we were able to pull that value downstream as it were and take advantage of that.

Thanks, everyone [Applause] END of Day-01

OpenStack Events

M2 instruct participants to: Open the Events section of the Contributor Guide. Read the material Ask The Mentor Questions Get Ready To Go Through The Exercises

OB1 Events and activities within the community
the aim of this exercise is to enable participants to be aware of activities that are happening within the OpenStack ecosystem.

1. Exercise 1

   M1 Lookup OpenStack(or OpenInfra) Days event close to where you live that you would be interested in attending. If there are no events planned, is there a meetup group? Share your response on the IRC channel. **OB1** The response in the IRC had a lot of mixed opinions depending on where each participant is located and if their locality is hosting local events or meetup groups. For example, 13 participants reported that they have no local events at all near their locality whereas 59 have one or more events reported. Also, all 72 participants reported that they will be willing to attend any event that is organized near their location.

2. Exercise 2

   M1 asked: " Look up the location and dates of the next PTG and share their response to the IRC channel." **OB1** All participants got the answer correctly from the OpenStack event page.

3. Exercise 3

   M2 instructed participants "Look up the location and dates of the next summit/Forum" **OB1** All participants got the answer correctly from the OpenStack event page.

   The general atmosphere of events happening was positive among participants and some participants registered for upcoming events near their local community.

DAY-02 Practical Activities

Survey Form

M1 and M2 reminded Mentors to be fully engaged

General Tips for Mentors before day 02 session

1. `Remain engage with the rest of the class even if you are not presenting.`
2. `Choose a table and sit with the students to help`
3. `If there aren't enough mentors or every table has one already, float around the room checking in on people, especially during exercises`
4. `When possible, sit at a table and build connections (networking) with participants`
5. `Talk slowly when you are presenting - English may not be their first language`
6. `Pause to ask students if they have questions on the material throughout your presentation`
7. `Ask the students questions to make sure they are engaged and understand the material`
8. `Join the IRC channel of the class and participate during the training`
9. `Give out swag and make sessions competitive`
10. `Promote ideas for next steps after training is done; mentoring, Project Onboarding and other related conference sessions`

The Lead mentors encouraged participants to be interactive

General Tips for Participants

1. `Use every opportunity you have to give us feedback. It's important for the community`
2. `Discuss your solutions with mentors and explain to them how you derived the solutions.`
3. `Use IRC for answering questions or the training etherpad if an exercise requires more space`
4. `Be prepared with the "deep dives" exercise, usually, participants have very different levels of knowledge and skillset.`

M1 presented the observer OB1 who also is a mentor in the OUI programs. OB1 told participants that if at any time they don't want to be part of the study they are free to quit without any consequence. Also, OB1 explains in brief what think-aloud is and that participants might be asked to think aloud when performing their tasks.

Seating configuration

Each table was arranged to accommodate up to six participants with two more reserve seats for mentors, making a total of eight people per table maximum capacity for all 12 tables arranged in the hall. These 12 tables formed 12 groups in such a way that table T1 was named group 1, …, and T12 was named group 12.

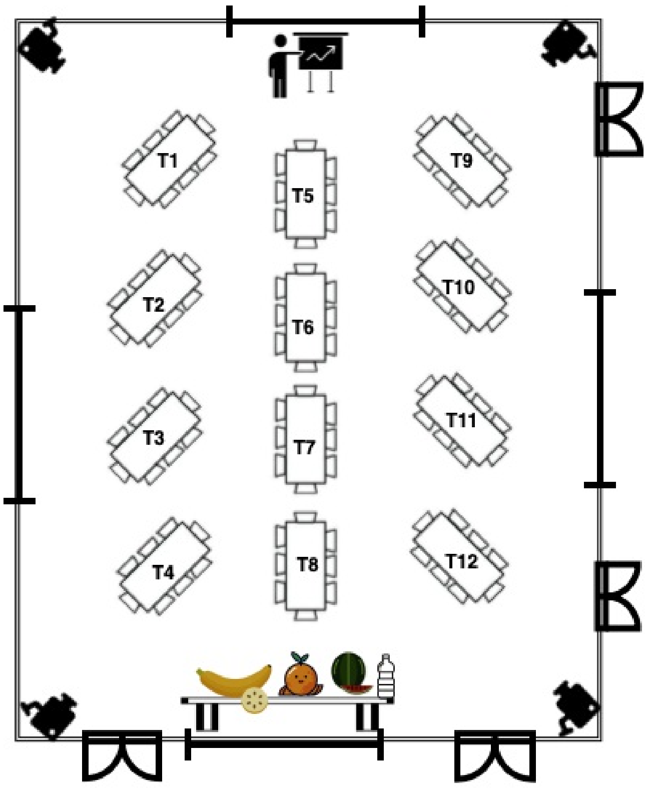

Figure 1: Setting of the observational study, participants were arranged in 12 tables named T1 to T12 respectively in 12 groups.

With this seating configuration, the observer (OB1) implore impression management skill in this direct observation study, which Erving Goffman proposed. OB collected field notes alongside the audio-visual recording. Meanwhile, the Hawthorne effect was high, that is the participants are totally aware that they are being observed. Also, the OB1 kept interaction with the participant tasks to the minimum.

Observable tasks

All participants were assigned the following tasks to walk-through with the assistance of mentors. In each task, OB1 randomly chooses a participant from a table and observes how they are performing the assigned task. Furthermore, OB1 asked participants to think aloud as they carry out their tasks.

Day-02 Agenda

(i) Overview of the contribution process (ii) Issues Tracking (iii) Git & Commit Messages best practices (iv) Code review process using Gerrit (v) Patch Gurus! (vi) Project Status and Zuul (Testing, CI & CD)

Hackathon (Dive deep code challenge)

Demo environment Code exercises

Learning Exercises

The following sections consist of hands-on exercises for participants to practice skills on contribution. Mentor **M2** motivated participants that in each series of exercises, the first person to finish and notify the mentors on IRC or on their table will receive a prize. There were varieties of prizes for everyone such as swangs, Lego, stickers, tickets for free summit outing events, etc.

Task1 ~~(i)~~

1. Overview of the contribution process

   M5 and **M11** presented a general overview of the contribution process, without going into much detail, which is reserved for later exercises.

   Getting to Know your project

   OB1 randomly chooses **P3** who was seated on table #1. **M5** told participants that "each project within OpenStack has its own purpose and culture." **M5** asked all participants to clone any of the OpenStack projects that they are **familiar with.**

   - $ git clone https://opendev.org/openstack/<your-project>

   M5 : Know who you are, your strength, weakness, and domain of interest. Based on that, select a project that reflects those interests "Get familiar with the codebase and programming paradigm of your selected project." **OB1** : "Which project have you chosen and why?" **P3**: "I chose Keystone based on what the mentors presented yesterday about the core OpenStack projects, and what I have searched so far, 'Keystone service provides API client authentication …' At school, I work with projects that implement identity authentication as services to web-based applications. So, I am most comfortable with this way of writing coding and thinking." **M5** : Explore and identify at least three functionalities in your selected project. In case you need help, the mentors assigned to your table are willing to help. **OB1** : While you are exploring your project of choice, can you think aloud? Explain your steps and what the functions do. **P3** : Now, I am searching the keystone documentation to see which functionality I am familiar with within the most recent release of OpenStack. I have Identified the first, functionality: A function that creates an OpenStack user with federated identity. Now, I am reading what the code does, … it takes four-parameter and returns a dictionary containing the user reference. Next, I am searching the keystone documentation again, …, I have found another functionality that returns a consumer and the consumer's secret. The search continues… In the manager class, I have seen a function that dynamically calls the backend. This function serves as the default pivot point for authenticating backends, and the search continues … Last, I have found a function that validates authentication from a query string. I will stop the search here because I fill satisfied with this task.

   - **M11** instructed participants to run different test cases in each project that they cloned. "If you need help, mentors are seated on your tables, they will assist you in running the test cases."
   - **OB1** noticed that "*P5* is having difficulties locating which test case to run, and asked the mentor assigned to their table for help."
   - **M3** asked **P5** "What is the problem, and how can I be of help?"
   - **P5** "I ran the commands exactly as **M11** instructed but having an error message."
   - **M3** "Let's re-run the command, and read the error message together."
   - **OB1** observed that **P5** omitted one argument parameter in the command, which aims at specifying the type of test to run; the "functional" parameter.

     ```
     $ tox -e py27 – zaqar.tests.functional
     ```

   - Socio-technical interacting with your project :: **M11** encourage participants to "get engaged in a project of interest and join the IRC channel, make sure you follow and participate in project-related mail threads in the mailing list. Also, attend regular meetings, and get your hand dirty by filing, fixing, and triaging bugs. Filing a blueprint/spec, Implementing a blueprint/spec."
   - Building your Persona ::

M5 Asked participants to go through the documentation of their project and see if they can spot and fix any typo etc. **M5** Then ask participants to look at code that other developers have written to be familiar with the style and comments used. " mentors are available to help if you don't understand what a code you are reading is doing. **M5** also emphasizes that one way to understand and grow in a project is to use the common forum/channel that developers used to ask/answer questions. **M5** then asked participants to join the project irc-channel(s) and make an attempt to direct people who are asking questions there to the right resources/documentation. "Make sure your mentor knows your interests in participating in the project. Explain why you are interested and what are your strengths, then ask how you can help the project."

M11 Build a network within your project team

> "Pay attention to who is an 'expert' in your project domain, don't forget to post and ask questions in the channel or send direct messages. Remember that If people know you, you have a better chance at your code getting attention."

General contribution workflow

M5 Gives the general contribution workflow, which consists of picking a task (this could be a bug, trivial fix, documentation, implementation), creating a new branch in your local repository, making the desired code change, adding and running test cases, last, create your commit and push the changes back upstream for review. However, **M5** told participants that "we will go into this later on in more detail so hold off on answering detailed questions for now."

Your patch upstream

M11 told participants that after pushing their changes upstream a CI/CD job will spin "Zull CI will review your patch." Community members with a +2 power will also review your patch. "Make sure you reply to the reviewer's comments on time and make requested changes then push back a new patchset. In some cases, you will have to handle merge conflicts."

Speeding Acceptance

M5 told participants to be consistent within and be on top of the reviewer's comments. Moreover, urge participants to be patient during the review period and be communicative and collaborative "Remember this is an open-source world! Things happen on the community schedule, not yours."

Task2 ~~(ii)~~

1. Issues/Task Tracking

   M7 and **M10**

   Learning Exercise
   Open the Task Tracking section of the Contributor Guide

   Read the material here [https://docs.openstack.org/contributors/common/task-tracking.html]

   Ask the mentors questions

   P23 asked a question to a mentor assigned to table T4, however, **M8** decided to share the question among all the mentors and the main coordinator **M1** asked that the question be repeated for the benefit of all participants. **P23** "Why is OpenStack using multiple task tracking systems? What are the main differences and drawbacks?" **M12** an expert in the ecosystem and PTL of the Storyboard project answered: " Originally, OpenStack used Launchpad as the issue tracker, Launchpad was developed and managed by Canonical to track bugs or blueprint. Moreover, Launchpad is limited in terms of scope. It was tied down to a project, it was not designed to support an ecosystem scope i.e a cross-project setting.

   However, Storyboard was engineered to support the coordination of cross-project work in an ecosystem setting, in which each project is different in the process of reporting bugs and planning new features, for example, a story could be to invent some new feature A, and tasks would be changed in project X, change in project Y, and change in project Z. Those changes need to merge in order to complete feature A. … Is that clear enough?” [Yes!]

Get ready to go through the exercises

OB1 observed that the mentors prepared the “material” to reflect the recent changes in the ecosystem codebase. All the projects involved were up to date. Moreover, participants on average could read and follow the instructions with little help from mentors, even though some sections were challenging, which we will detail later. OB1 move randomly from Table 1 to Table 8 and randomly chose **P40**.

1. Exercise 1

   OB1

   > The aim of this exercise is to enable participants to practice how to create, report, assign and fix, review a bug, knowing a bug life cycle is important in contributing to OpenStack.

   Mentor **M10** asked the participants to “create and submit a bug to our sandbox repository. This should include at least one tag when creating your bug. Once it has been created, assign it to yourself.” **OB1** asked **P40** to think aloud and explain each step as they proceed if possible. **P40** First, I am using my favorite editor vim to create a python file. I will call it exercise1.py Next, I am writing a function that reads the prints of all OpenStack summits and their locations in the past [… goes silent for a while…] Now, I am injecting a bug to my code with the tag Bug101. This bug is assigning the wrong locations to each summit. I am done with the code. I am adding it to my stage area, … Now, pushing it to the sandbox repo … done!Now, I am signing in to launchpad … I am reporting the bug now and assigning it to myself … the bug is now assigned to me. **OB1** noticed that **P33** and **P35** seated on table/group 10, were exchanging ideas constantly throughout this exercise 1, therefore, **OB1** moved to table 10 and asked both **P33** and **P35** how they found the exercise and if they could walk him through the steps that they took in doing the exercise. **P33** said “this was my first time working with git. At school, I did mostly theoretical computer science and mathematics, I know the logic and algorithm behind most code but have not been exposed to real situations. So it was relatively hard to work along, but **P35** seems to have a better experience with the git version control system. However, **P33** affirms that the concept of what the exercise demands is not that complex to understand, except the technical knowhow to get it done.” **P35** added that “ I used git a lot at college in nearly all my software engineering courses and projects, therefore, I find this exercise pretty straightforward. Except for the launchpad thing that I am using for my first time today, but overall, the exercise is not that hard for me.” **P33** Concerning the code, we implemented a basic search algorithm in python and injected a bug to the code that always returns the first element. I used SPE editor on my Ubuntu machine to write the code and then **P35** used the command line with git to commit the changes to the sandbox repository. After that, we navigate the interface of Launchpad to understand how it works, then, I assigned it to **P35**.

   OB1 observed that

   > participants within each group were sharing information to help solve their problems and also using the IRC common channel to share their ideas or ask questions from other participants outside their group/table. For example, **P44** asked a question to **P40** how to assign the bug on launchpad, and **P40** showed **P44** the steps to accomplish that task. Moreover, similar kinds of collaboration were happening across the different groups/tables among participants. Also, **OB1** noticed that participants were asking questions on the IRC channel and other participants from different groups/tables were sharing their answers to the channel.

   M10 told participants that, “You will use this code later when we practice writing commit messages and pushing patches to the sandbox repo so make it interesting!” https://bugs.launchpad.net/openstack-dev-sandbox/+filebug

2. Exercise 2

   M2 Blueprints are used to track the implementation of significant features in OpenStack. Keeping their status current is critical to the success of the release and the project as a whole.

OB1

> The aim of this exercise is to enable participants to practice how to create and register new features commonly known as a blueprint. Blueprints are artifacts that enable the growth of the ecosystem in terms of functionalities.

M12 instructed participants to "create and register a blueprint against the sandbox repository. Include a description and assign yourself." Specify the Name, Title, and description of what the blueprint should accomplish

You will use this blueprint later when we practice writing commit messages and pushing patches to the sandbox repo so make it interesting!
https://blueprints.launchpad.net/openstack-dev-sandbox

M2 "Read and follow the online instructions on the blueprint. Once you finish your task don't forget to indicate on the IRC channel and the motors will verify your blueprint, the first person will get a reward." **OB1** moves to table/group 5 and observe how six participants **P25**, **P26**, … **P30** are creating and registering their blueprint. **OB1** observed that these six participants were mostly working independently on this task. **OB1** also monitors the IRC channel to record the first person to finish the task and how long it takes. Moreover, the participants were not required to write a concrete blueprint comparable to those that have been implemented. But, they should just follow the right procedure and respect the norms, which were presented to them on Day-01. All participants registered their blueprint on launchpad following this template on the sandbox repo: Register a blueprint in Launchpad to their project page at launchpad.net/$PROJECT and clicking "Register a blueprint" Enter blueprint Name, Title and Describe the feature summarily in the blueprint itself Participants: Link to another document (using the specification link) if you have more Set assignee <<participant >> Mentors: Approve/Reject the blueprint and provide feedback/comments.

OB1 noticed that the first participant to create and register a blueprint is participant **P13** and it took 19 min tho do so, mentors gave **P13** a sticker. Meanwhile, the last participant finished in 27 min.

P27 created an elaborated blueprint that aims at provisioning NAS services to facilitate file storage. The description of the blueprint was two paragraphs long, which was not easy to understand. No external link was referenced to this blueprint. **OB1** asked to share their experience on this exercise. **P27** "It was a straightforward exercise, I wrote my blueprint for the Cinder project because Cinder is responsible for block storage at OpenStack and that is what interests me most, at least for now, but the task required writing skills that I have not really developed. I am still struggling with my writing skills. So, it took me a long time to write the summary of the blueprint"

Mentors rejected all the blueprints and then after round(s) of reviews, all were finally approved. **M2** stated that "the decisions to reject or to approve your blueprint were taken based on learning purpose only and not on the technical relevance. We wanted everyone to get familiar with the process involved rather than paying attention to the actual specifications. You might have noticed that the feedback that mentors provided were actually the writing approach they expected you to write specifications and that is the best practice that we encourage. "

3. Exercise 3

M8 told participants that: "Now, go and post comments on a bug(s) or add some ideas on a blueprint's whiteboard that was created by someone in your group. For example, You can ask a question about the issue or proposed feature. You can confirm the issue and update its status to triage."

OB1

> The aim of this exercise is to enable participants to practice and develop the skill of writing and reviewing standard bug's comments (+/- 1 when necessary) on Launchpad. This activity of writing standard bug's comments is of critical importance in the OpenStack ecosystem and is highly encouraged.

M3 Don't be afraid to make mistakes, try your best to write clear and concise sentences to explain what you are trying to achieve. Don't forget that reviewers will take a look at your work and give you feedback. This is one way of learning the way things are done at your project level.

Also, once reviewers notice you are making efforts even when you make mistakes they will reach out to help you out perfect your skills. So, read code experts have written and learned from their mistakes, how the review process helped their patches to get better and accepted.

and register new features commonly known as a blueprint. Blueprints are artifacts that enable the growth of the ecosystem in terms of functionalities.

OB1 Noticed that all 12 group participants were paired 2-by-2 to work on this exercise. In some groups for example group 1, **P1** post a comment on **P2** bug and **P2** post a comment on P1 and the rest in the group. On the other hand, in other groups such as group 7, **P37** post comment on P38, P38 post comment on P39, … P41-42, and P42 post a comment on P37, etc and on the blueprint, they reversed the order.

4. Exercise 4

   OB1 moves to group/table 2.

   This exercise is similar to the previous lab; exercise 3 on Launchpad (individual bug tracker), but now, on Storyboard (cross-project bug tracker).

   M11 instructed participants to "create a board with at least two worklists (one manual and one automatic) for organizing stories you are going to create in the next exercise. For the automatic worklist, give at least two criteria for the items that will go into the worklist. These criteria can be matching a project-group, story tag, etc." https://storyboard-dev.openstack.org/#!/dashboard/

   OB1

   > The aim of this exercise is to enable participants to practice and develop the skill of writing and reviewing standard bug's comments (+/- 1 when necessary) on Storyboard. Participants will appreciate the difference between Launchpad (individual project level) and Storyboard (ecosystem level), and how cross-project bugs are handled, which was a nightmare on Launchpad.

   M12 gives a brief explanation on Storyboard "In Storyboard, a story is a bug report or proposed feature. Stories are then further split into tasks, which affect a given project and branch. Thus, contributors can track their work efficiently across several interrelated projects, which was impossible with Launchpad. For categorization or prioritization, stories and tasks can be gathered in ordered worklists. Teams, projects, or sponsors may create a board with manual or automatic lanes to provide a clear overview of the activity of interest."

   M1 called the attention of participants and told them that "I will show a live demo on how to use a storyboard, the different tabs and widgets that you will use in this exercise, and how storyboard works in general."

   OB1 observed that all the participants were able to follow the instructions and example given by **M1** and did the first part of this exercise between 10 - 15 min, however, the second part, which required participants to automate a worklist, participants spent on average 23 mins to complete the task. The first participants to finish this exercise was **P58** in group 10.

5. Exercise 5

   M10 instruct participants "create three stories for your worklists. At least one of them should meet the criteria that enable it to appear in your automatic worklist. Each story must have a task that is named differently than the story name. You will use these later when we practice writing commit messages and pushing patches to the sandbox repo so make them interesting!"

   OB1

The aim of this exercise is to enable participants to be familiar with a cross-project task tracker for bugs and features, and to automate tasks.

OB1 moves to group 11/T11, and observe participants **P61 - P66**. Members of this group were collaborating and exchanging ideas among themselves, but each member created their task independently. For example, this group deliberate and came up with an architecture of this exercise. On the one hand, they design a worklist, and this worklist defines the state of these three stories to either be manual or automated. Then, they divide the stories into two groups; automatic and manual. Moreover, they assigned tasks for each story.

Each member of group 66 then implemented this design individually. For example, **P62** defines criteria for all his stories to be automated, meanwhile **P61** and **P64** defined exactly one to be automated. **P63**, **P65**, and **P66** defined two criteria for automation. However, the way group members named their stories and tasks were distinct.

6. Exercise 6

   - **OB1** moves to group 12/T12 and observes how group members **P67 P72** were collaborating among themselves.

M3 told participants that "share your board with your group and assign yourself to two tasks on other people's stories. Comment on one story."

OB1

The aim of this exercise is to enable participants to learn how to collaborate and exchange knowledge, and to practice how the review process is coordinated in a team.

M8 said " write down your question on a group member's comment, if you have a question. and not to do so verbatim. Writing down comments will improve their communication skills and make things clearer for both the contributor and the reviewer.

- **OB1** observed that participants were more committed to giving feedback and asking questions on group members' board than the time they spend writing their stories.
- **OB1** asked group members how they found this exercise and what is most exciting about it?
- **P70** said: "I realize that reviewing the work that my teammates have done makes me see contributing to a project differently. For example, I was limited to my own ways of thinking but now I realize that when I read a teammate's logic and get lost, I know exactly where I don't understand and I ask questions for clarification. Also, I have learned something new that I did not know. "
- **P67** said: "In my experience, I noticed that I can easily find something wrong on what someone has done rather than seeing something wrong on my code or what I have done. The review process stands out as most exciting for me because my critical mindset was more activated than just focusing on my own work, I try to see things through the lens of what someone has done to make sense out of it, that is exciting."

Task3 ~~(iii)~~

1. Git & Commit Messages

M4 said: "Git is an important tool that you will need all your life in the world of open/closed-source to contributing to any community's codebase. In this task, we will walk you through the basics of Git, which is fundamental to contributing to any project, but the hard part of the work remains in you to practice, practice and practice until you are comfortable."

OB1

The aim of this section is to equip participants with the required skills using Git to contribute to OpenStack as a contributor.

M9 instructed participants: "Open the Setup and Learn GIT section of the Contributor Guide, read the material, ask questions to the mentor, and get ready to go through the exercises." **OB1** Participants are busy reading the study guilds on Git

Git best practices, we recommend the following structure in Commit Messages

M9 "Commit messages are the first things a reviewer sees and are used as descriptions in the git log. They provide a description of the history of changes in a repository. Commit messages cannot be modified once the patch is merged." Structure: Summary Line Empty line Body Empty line Tags

OB1 observed that **M4** shows a slide with each line and a detailed description

Summary Line The summary line briefly describes the patch content. The character limit is 50 characters. The summary line should not end with a period. If the change is not finished at the time of the commit, start the commit message with WIP.

Body The body contains the explanation of the issue being solved and why it should be fixed, the description of the solution, and additional optional information on how it improves the code structure, or references to other relevant patches, for example. The lines are limited to 72 characters. The body should contain all the important information related to the problem, without assuming that the reader understands the source of the problem or has access to external sites.

Tags Tags are references used to link the change to other tools. For example, the "Change-id" line is a unique hash describing the change, which is generated automatically by a Git commit hook. This should not be changed when rebasing a commit following review feedback, since it is used by Gerrit, to track versions of a patch.

Read the materials for other used tags at OpenStack.

M4 Now, the following exercises will help you practice each line at a time until you get a complete well-structured commit message. Remember that the mentors are here to assist you.

1. Exercise 1

   M4 "write a summary line for each bug, blueprint, and story you created during our task tracking exercises. Share them on our IRC channel."

   OB1 is monitoring the activities on the IRC channel to see what participants are sharing and observed that "Mentors were commenting on participants post. Those that were not written properly on the first try (54/72) got a (-1) and feedback from a mentor stating why it's not good, whereas those that we accepted got a (+1) and a reviewer's comment to encourage the effort."

   In the second round, those that got a -1 had the change to implement the reviewers' comments and improve their text summarily. Thus, 53/54 got a +1 except one participant (P39) who still got another -1. This time, mentor **M6** approached **P39** and asked if they need some help summarizing their text to give meaning to the changes made? **P39** affirms needing help and both M6 and P39 worked together to write and acceptable summarized text.

   M6 gave general feedback to participants on how to catch reviewers' attention with a good summarized text that explain your code change and posted some good and bad samples on the IRC channel.

2. Exercise 2

   M4 instructed participants: "write the body of a commit message to expand on the summary lines you just wrote. Feel free to make up details to make the context more realistic, then share them on IRC."

   OB1 moved to table/group 8 while monitoring the IRC channel and observing how participants were asking questions to mentors and communicating among themselves in their groups.

OB1 asked **P46** to think aloud while performing the task. **P46** "I am writing the body to have all the essential elements […], I have 120 characters I am now worrying about reducing the number of characters to 72 [… thinking …] let me play with words a bit to see what I get here […] The best I can do is 98 characters.

I am now posting this body to the IRC channel … waiting for feedback

Meanwhile **P43** and **P47** submitted 65 and 87 characters respectively on the IRC channel.

On the IRC channel, other group members have submitted their body text awaiting review. 21 participants respected the 72 characters recommended length. Meanwhile, all the 72 participants submitted their text waiting for reviewers' comments.

All the mentors are now writing their reviews online, communicating their comments to the participants directly on the IRC channel and everyone could see and learn from the mistakes/strength of others.

M2 "Besides the character length constraint, overall, the content was meaningful. All the body text carefully respected the recommendation."

All the 21 participants who respected the body text limit got a +1 and the others -1, and the first participant to post their text was rewarded with a sticker. Moreover, **M1** said "We applied strick measures here to make sure to get this right and respect the standard because one major area in OpenStack that makes code review difficult is the commit message. Make sure you work with mentors to get this right and we move on to the next activity."

Mentors collaborated with the 51 participants and get the body text to match the limit of 72.

3. Exercise 3

   M8 instructed participants "put the pieces together and finish your commit message! Make sure to include the summary line, body, and the required external references along with any optional external references you think it may benefit from. Then, share the commit message with someone sitting next to you. Give them feedback on their commit messages."

   OB1 moved to table/group 10. Participants used the feedback from mentors and added a tag to their commit messages and reference links as required. In this exercise, all the participants did as expected and they were able to learn the peer-review process.

   P54 said, "The exciting thing in this task as I see it is sharing what we learn from the mentors, as we use those skills to give feedback to our peers and also learning from them."

   P60 "I am fascinated with the comment of my teammate **P59**, based on the other peer review exercise we did earlier and now, I see that the mentors' feedback has some influence in what he wrote, because the feedback is almost identical to the feedback that the mentor **M10** gave us."

Task4 ~~(iv)~~

1. Code review process using Gerrit

   M10 "Make sure you configure your Gerrit account: open the Setting Up Your Gerrit Account section of the Contributor Guide. Read the material and ask questions to the mentors, then get ready to go through the exercises."

   M10 explain to participants that Gerrit is the review system that the OpenStack community uses. Gerrit allows contributors to Get reviews on contributors changes proposed to OpenStack repositories Request reviews from specific community members Make quick changes to your patches in the WebUI

   OB1 Getting familiar with Gerrit
   the aim of this exercise is to enable contributors to get familiar with the Gerrit code review tool.

1. Exercise 1

   M8 asked participants that: “how do you initialize your local repo with git review? Write your response on the IRC channel” **P3** responded “ git review -s”, and **M2** rewarded **P3** with a sticker **M1** added Git review is a tool maintained by the OpenStack community. It adds an additional sub-command to ‘git’ such as git review.

2. Exercise 2

   M8 What does ICLA stand for? What is it? Write your answer on the IRC channel **P55** responded first and said: “ICLA stands for ‘Individual Contributor License Agreement’ this is a formal agreement, which protects intellectual property rights granted with contributions from a person or entity.”

   M1 gave a Swang to **P55**

   M5 instructed participants that “Following is a list of the commands that you need to know for your first contribution.”

   To clone a copy of some repository.

   git clone https://opendev.org/openstack/<PROJECT$_{NAME}$> After you’ve completed the Setup and Learn GIT section, the following command configures the repository to know about Gerrit and installs the Change-Id commit hook. You only need to do this once per repository you cloned:

   git review -s As **P3** rightly stated. To create your development branch (substitute branch$_{name}$ for a name of your choice). It’s better to create a new branch for each patch than working from master: git checkout -b <branch$_{name}$> To check the files that have been updated in your branch: git status To check the differences between your branch and the repository: git diff master Assuming you have not added new files, you commit all your changes using: git commit -s -a Read the Summary of Git commit message structure for best practices on writing the commit message. When you are ready to send your changes for review use: git review If successful, the Git response message will contain a URL you can use to track your changes. If you need to make further changes to the same review, you can commit them using: git commit -a –amend This will commit the changes under the same set of changes you issued earlier. Notice that in order to send your latest version for review, you will still need to call: git review

3. Exercise 3

   M7 instructed participants “Review three patches in the sandbox environment: https://docs.openstack.org/contributors/code-and- documentation/sandbox-house-rules.html Try to find things to make comments on even if they are just asking a question and not pointing out an issue, don’t just +1 three different patches.” Invite at least two mentors to review your work. **M8** added that participants should consider applying all the skills they learned earlier and use them in this activity **OB1** observed that participants were submitting their changes this time with complete and correctly formatted commit messages. The use of voting was also appropriate +/- 1. Overall, all participants submitted their works and got different feedback from mentors.

4. Exercise 4

   M5 Asked participants to build their persona on their chosen project or cross-project. Review other’s code, help fix the documentation, answer other’s questions or help direct them to those who can. Let a mentor know you are interested in participating in the project Explain why you are interested and what are your strengths ask how you can help the project

   OB1 observed that this exercise was challenging to participants even though they were excited about the challenge. For example, **P19** said “I would have chosen to review a code that someone else wrote, but, I have the feeling that I am not yet ready. What if I get it wrong and the person is more experienced than myself? What if the code is more complex than I perceived? …, I will prefer to go with documentation or better still helping someone who has a question on directing them to the right resource.”

P41 “Maybe I am wrong but working on documentation seems difficult for me because I don’t think my language skills are good enough. I can unintentionally make more typos than fixing them :) I will prefer more coding …”

Task5 ~~(v)~~

1. Patch Guru

   OB1 How to Become a Patch Guru?

   When you are working on implementing a new feature or adding documentation to an already existing one it is easy to get carried away by the work itself and forget about the unwritten rules of constructing your changes.

   M9 explained to participants that this section will guide them on how to create patches that people will want to review. Moreover, it will enable you to know how to structure a patch that makes it easier to maintain throughout the review process, and how to structure a patch that is easier for community members to review.

   Learning Exercise

   - Open the How to Become a Patch Guru? Section of the Contributor Guide
   - Read the material
   - Ask Questions to The Mentors
   - Get Ready To Go Through The Exercises

   M9 gave some recommendations concerning patch size: “Reviewing large patches is very inconvenient and time-consuming therefore we always suggest breaking down your changes into smaller blocks. While there is no magic number, try to keep the size of your changes as small as possible, but under a few hundreds of lines changed total. Patches that test heavy with little code change require as much effort as code-heavy changes.”

   M10 remember that “Longer patches require more time to review; wherever you can, keep the length reasonable. And where you can’t, you can help the reviewers by adding code comments and writing a detailed commit message to describe the changes you introduced in your patch.”

   1. Exercise 1

      M6 How do you handle dependent changes in the same or multiple repositories? **P12** posted the answer on IRC: “When you have changes in multiple project repositories you can mark dependent patches with the ‘Depends-On’ tag. The tag will appear as a link in the commit message which helps you and also the reviewers to track and navigate between the dependencies of your changes. The ‘Depends-On’ tag is a marker on your changes and when used a patch cannot be merged until all its dependencies are landed.”

      M3 Reviewed the response and gave **P12** a sticker and added that “The tag can be applied to patches proposed for the same repository as well. In that case, the changes are separate enough to be kept independent which means that if you need to fix changes from review comments you can do it on a per patch basis. It is also true for rebasing each patch.”

   2. Exercise 2

      M5 asked participants that “What command do you use to modify a patch within a chain?” **M10** added that “A patch chain is easy to handle if you keep in mind a few recommendations: Always have a local branch for these changes to ensure that you don’t mix it together with changes related to another feature or bug fix. Always handle a chain as one block of changes by rebasing the whole chain and keep it up to date when you modify a patch to fix review comments or add changes to it.”

P31 answer the question that **M5** asked that “o modify a patch within a chain you will need to use interactive rebase” git rebase -i HEAD^

M11 added that “You need as many ‘^’ as the number of the patch you want to edit first from the top of the chain. Alternatively, you may wish to use git-restack, which figures out the appropriate git rebase command for you.”

OB1 both responses from participants we recommended by mentors and they were rewarded with stickers.

Task6 ~~(vi)~~

1. Project Status and Zuul (CI & CD)

M4 instructed participants to “Open the CheckingStatusinZuulsection the Contributor Guide Open the Using Elastic Recheck section of the Contributor Guide. Read Material, ask questions to the mentors. Get Ready To Go Through The Exercises.”

OB1 Monitoring your patch on Zuul (CI/CD)
The aim of this activity is to enable participants to get accustomed to the CI/CD system at OpenStack, OpenStack uses Zuul CI/CD.

M4 said “Zuul is a CI/CD tool developed and maintained by the Infrastructure team at OpenStack. Zull provides OpenStack projects and cross-projects a means of defining test jobs, which runs on each proposed commit. These tests must pass before any patch can merge. Therefore, once a contributor pushes a patch to gerrit, zuul will automatically trigger jobs to verify the patch functions properly.”

M4 Let’s take a look on how to track jobs submitted to Zull to see their status go to this link: https://zuul.openstack.org/ **OB1** observed that as **M4** explain in detail how the graphical interface of Zull works, showing the different functionalities, participants we carefully following and some were taking pictures of the screen and some videos to capture the moment of interest.

M6 told participants that “after understanding the Zuul job status and how gating works, we will now move to ‘Elastic Recheck’ that enables contributors to (i) Enhance the automatic testing, which OpenStack community encourages and enforced on every patchset that is submitted to gerrit. (ii) Report recurring bugs so that you don’t need to manually ‘recheck . ”

M6 explained in detail over slides and video what Elastic Recheck is and how it works: “Elastic-recheck is a tool used to track failures in test jobs. Elastic-recheck is built on top of an ELK (Elastic Search, Logstash, Kibana) stack where we use Logstash to store all logs from CI jobs in an ElasticSearch cluster … ”

P63 asked the mentors that “should we master how these technologies work before running jobs on Zuul?”

M7 answered that “Not really upfront because the system has bee configured already by the infa team, but it's good to know what each stage is doing to be an expert. This also comes with time. I know that, It took me over 4 years to have a good mastery of how everything fits perfectly together and I still learn everyday. So, …”

1. Exercise 1

M5 opened the Zuul status page and instructed participants to “Look at the Zulu status page Find Information That Can Be Retrieved Foreach patch in a pipeline What happens if you click on a patch under test How Many Gate And Check Jobs Are Running Pick your favorite project and report how many jobs has running in IRC What Is The Significance Of The Dots, lines and colors Discuss Your Findings With Your Group” **OB1** observed that all the participants were flooding the IRC channel with responses and the mentors were given feedback immediately to each response as they came in. Moreover, on the fourth point, **M9** highlights that all participants should mention the project that the pick alongside the number of jobs that are running. **P2** said that they have noticed that the jobs on Zuul are running real time and changes every time, so the values they are reporting may change when the mentors want to verify their responses. **M10** recommended **P2** observation and explain that that is the reason why the last point is asking them to discuss their findings with their respective groups.

OB1 To demonstrate, **M1** walk through all the steps and explain to the participants what happens at each stage.

2. Exercise 2

M6 instructed participants to "find how Checks are categorized and discuss with your table how you would Determine that you have encountered one of these bugs?" **OB1** to further simplified the task, **M6** walk participants through a serie of video explanations to show classification in rechecks.

All the participants watch and follow the video explanation and discuss how they will determine bugs in recheck.

P51 said "I think I am now getting more confidence with my understanding on this Zuul ands recheck, especially when **M6** explain a few minutes ago, that was a great explanation!"

P29 on his part said: "I think the load of materials has been too overwhelming but the mentors are making it looks too easy for me to follow the concepts."

Each topic in this agenda for **Day-02** follows a series of hands-on exercises that the mentors ask participants to do, with an exception of the hackathon that exposes the participants to critical thinking and self-developed skills. There is a lunch break between 12-00 pm and 12-30 pm, with a short break interval of 5 mins after every Task.

Task7 ~~(vii)~~

1. Hackathon (Dive deep code challenge)
   - Demo environment
   - Code exercises (Testing)
   - **OB1** :: Demo environment → Running OpenStack deployment through DevStack documentation:https://docs.openstack.org/devstack/

M8 Instructed participants to spin up DevStack in the local environment, and understand OpenStack cloud services to do the code challenge.

```
git clone https://opendev.org/openstack/devstack
cd ./devstack
cp ./samples/local.conf . vim ./local.conf
```

M8 instructed participants that "Go to: http://localhost/ and access the horizon portal Follow the step-by-step instructions in your material guide documentation and issue all the commands."

OB1 observed that participants were busy throughout, reading documents and practicing the commands. They were also asking questions to the mentors on the IRC channel. Some example questions are: **P6** "Can I deploy DevStack to the cloud? Since it pulls all OpenStack services" **M7** answered that based on the documentation, it is stated that DevStack is only used for testing services and in development, but can't be used for production because once you shutdown DevStack, it destroys all the VMs.

M41 Asked that "Should all the services of OpenStack be up and running to use DevStack?" **M3** answered "No, you can run only the services that you want to test, or experiment with."

OB1 some participants were experimenting with the CLI meanwhile others were using the GUI to interact with VMs and services.

OB1 asked some participants that were using CLI and GUI to know the reason why. **P34** said: "I have always been more comfortable with commandline doing stuff. For example, at schools, I usually do most of my work with the terminal mode, such as Weka, Java, Python, and many more, it's more of a culture to me." Meanwhile, **P70** said: "I am more of a visual person and when learning something new for the first time, I like to see the interface and how it behaves , that is just me."

1. Exercise1 ~~Code exercises (Testing)~~

   DevStack exercise: **M9** instructed participants to "Start DevStack in a VM on your laptop or public cloud. Make sure that the services are running. Choose a service and issue an API call or use its client to verify functionality" **OB1** observe that participants that are more familiar with GUI used interfaces app to make API calls, whereas, CLI users call it from the terminal. Also, mentors were actually demonstrating how things work by example. Making it easy for participants to follow and understand the operations. LOG message exercise **M4** "Add a few extra LOG.debug() lines to one of the methods of the API call you chose in the previous exercise. Restart the corresponding service in your DevStack environment and find the new message in the logs. Find out what parameters were passed to that method by using the LOG messages" **P46** Asked the mentors "what level of logging should we use?" **M11** responded that "Good questions, you can use DEBUG and INFO and observe how both work, beside you can modify the code and try with other levels. It's up to you, but the default will be INFO." **OB1** observed that participants were practicing without the help of mentors fewer than three questions were asked by participants on the IRC channel and within the individual groups.

   - Testing
     - Testsuites Unit
       - Functional
       - Integration
     - Testing Framework
       - Tox

2. Exercise2 ~~Group Challenge~~

   OB1 Collaborating in Ecosystem projects
   the aim of this section is to introduce participants on collaboration, how to work in a team project using divide and conquer technique.

   M10 In this Exercise run the test suite with the tox framework. **OB1** observe that participants divided their task into three groups, One group ran one test suit and the group shared their knowledge and explain how all the test cases fit together. Mentors were mostly providing only guidance rather than directly helping participants to solve their problems. In most cases, mentors questioned the rationale of participants rather than answering their questions directly, this guided participants to think deeply and figure out their own solutions. **M5** Mentors in each group will break the tested code of one test case and in your group, you are allow to find the modification that they did by running the test and analyzing the test output **OB1** In each group the mentor of that group broke the code that participants tested above and asked the participants to identify and break in the code, after running the test case. Different groups apply different test cases and methods of analyzing the test output. However, all the 12 groups were able to identify the broken code. Hackathon: **M1** Find an open review, which is less complex and download the patch. Remove the code changes and run the tests Check whether the tests failed or not Explain what it means if they didn't Comments on open review

   OB1 real world contribution use case

   Participants are asked to go on the Gerrit system of OpenStack and practice with a code that has been submitted ready for review. Mentors are not required to provide assistance in this task. Until the end if a group didn't do it right, mentors can then provide guidance.

   OB1 observed that all 12 groups used different functionalities and projects. Moreover, Some groups spent time reading and understanding the commit messages before they started doing the exercise for example, Groups 2, 4, 5, 9,11, 12, whereas the other groups only focused on the sections of the code that was modified.

Also, **OB1** was walking around to see how different groups were working to understand the ways in which each group approached the problem.

Group members spend much time deliberating on their approach and solution than implementing the solution.

Participants were required to use all the materials that they have learned so far to solve this problem within a 25 min.

In the end, 10 groups submitted their solutions and got feedback from the mentors, except two groups (1 and 6) that were not able to submit a complete solution on time. Moreover, group 2 was the first to submit their solution.

Finally, after doing through all the groups submission, group two was declared the winner and all group members were awarded a ticket for a banquet with swangas each. The mentors then gave final remarks and encouraged participants to make good use of the mentoring program, which is the next step available free for those who are interested in. A group photo and refreshment closed the event.

OB1 asked participants to drop in a few words on the IRC channel what activity they like most and which they didn't like?

Likes: **P48** : "I like the hands-on section most and, of course, the sticker prizes!" **P1** : "I admired the explanations of different projects and how the form an ecosystem" **P15** The testimony on mentoring was great! I love it. **P6** "The CI/CD and testing sections was my favorite, I didn't do something like that at school" **P5** "The documentations were well structured and elaborative" **P31** "Mentors were great inspirations and knows their stuffs well" **P40** "We need more exercises like the hackathon, besides the mentors were great!" **P62** "" Dislikes: **P13** "Remove the events sections complete waste of time, my opinion though" **P4** "There are lots of account to create, this can be very confusing" **P2** "Lots of things to master in a short period of time"

Technical Details ~~Guidelines~~

Debugging Code
    LOG.debug()

M4 Thought participants techniques how to debug their source code and how debugging applies to Projects.

1. Technical Activities during Day-02

   All 72 participants were actively participating in the coding activities that the mentors assigned them with.

2. Writing and maintaining Quality Code

   1. Git (best practices on git **Commit messages**)

      M7
          gave detailed layout structures of commit messages that respect best practices.

      M7 stated that: "Based on many years of practical experiences doing code development, bug troubleshooting and code review across OpenStack projects and other communities such as Linux kernel, CoreUtils, GNULIB, etc., we suggest a fairly common practice, which is motivated by OpenStack strong desire to improve the quality of it's projects' Git histories."

      M7
          "… We will demonstrate the benefits in splitting up **changes** into a sequence of

      individual commits, and the importance in writing good commit messages to go along with them."

- M7 divided the topic of commits in two sub topics (A/B), and gave the advantages in splitting commits.
- The structured set/split of the code changes "If a code change can be split into a sequence of patches/commits, then it should be split."
- The smaller the amount of code changed, the easier it is to review & identify potential flaws.
- If a change is found to be flawed later, it may be necessary to revert the broken commit. This is much easier to do if there are not other unrelated code changes entangled with the original commit.
- When troubleshooting problems using Git's bisect capability, small well defined changes will aid in isolating exactly where the code problem was introduced.
- When browsing history using Git annotate/blame, small well defined
- changes aids isolates exactly where & why a piece of code came from.

```
commit 3114a97ba188895daff4a3d337b2c73855d4632d [ID-1]
 Author: [removed]
 Date:   Mon Jun 11 17:16:10 2012 +0100

   Update default policies for KVM guest PIT & RTC timers

 commit 573ada525b8a7384398a8d7d5f094f343555df56 [ID-2]
 Author: [removed]
 Date:   Tue May 1 17:09:32 2012 +0100

   Add support for configuring libvirt VM clock and timers
```

Furthermore, **M7** explain these two commits, which provide support for configuring the KVM guest timers. The introduction of the new APIs for creating libvirt XML configuration have been clearly separated from the change to the KVM guest creation policy, which uses the new APIs.

1. The information provided in the commit message

M3
: highlight some key points when writing a good commit message

Reflect on these points whenever you are about to write a commit message:

1. Do not assume the reviewer understands what the original problem was.
2. Do not assume the reviewer has access to external web services/site; The commit message should be totally self-contained, to maintain that benefit.
3. Do not assume the code is self-evident/self-documenting.
4. Describe why a change is being made.
5. Read the commit message to see if it hints at improved code structure.
6. Ensure sufficient information to decide whether to review.
7. The first commit line is the most important.
8. Describe any limitations of the current code.
9. Do not include patch set-specific comments.
10. Always use the appropriate tags; Change-id, bug-ID, etc.

Summary of Git commit message structure

- Provide a brief description of the change in the first line.
- Insert a single blank line after the first line.
- Provide a detailed description of the change in the following lines, breaking paragraphs where needed.
- The first line should be limited to 50 characters and should not end with a period.
- Subsequent lines should be wrapped at 72 characters.

```
$ git commit -s [--signed-off-By:]
```

Example of a good commit message shown to participants

```
commit 3114a97ba188895daff4a3d337b2c73855d4632d                  (1)
  Author: [removed]                                              (2)
  Date:   Mon Jun 11 17:16:10 2012 +0100                         (3)
                                                                 (S)
    Update default policies for KVM guest PIT & RTC timers       (4)
                                                                 (S)
    The default policies for the KVM guest PIT and RTC timers    (5)
    are not very good at maintaining reliable time in guest
    operating systems. In particular Windows 7 guests will
    often crash with the default KVM timer policies, and old
    Linux guests will have very bad time drift
                                                                  (S)
    Set the PIT such that missed ticks are injected at the       (6A)
    normal rate, ie they are delayed
                                                                  (S)
    Set the RTC such that missed ticks are injected at a         (6B)
    higher rate to "catch up"
                                                                  (S)
    This corresponds to the following libvirt XML                 (7A)
                                                                  (S)
      <clock offset='utc'>                                        (7B)
        <timer name='pit' tickpolicy='delay'/>
        <timer name='rtc' tickpolicy='catchup'/>
      </clock>
                                                                 (S)
    And the following KVM options                                (7C)
                                                                 (S)
      -no-kvm-pit-reinjection                                    (7D)
      -rtc base=utc,driftfix=slew                                (7E)
                                                                 (S)
    This should provide a default configuration that works       (8)
    acceptably for most OS types. In the future this will
    likely need to be made configurable per-guest OS type.
                                                                 (S)
    Closes-Bug: #1011848                                         (9)
                                                                 (S)
    Change-Id: Iafb0e2192b5f3c05b6395ffdfa14f86a98ce3d1f         (10)
```

- (S) -> White Spaces
- (1) -> Commit ID
- (2) -> Author ID
- (3) -> Date
- (4) -> change request subject (first line of the commit message)
- (5) -> describes the original problem (bad KVM defaults)
- (6) -> describes the functional change being made (the PIT/RTC policies)
- (7) -> describes what the result of the change is (the XML/QEMU args)
- (8) -> describes scope for future improvement (the possible per-OS type config)
- (9) -> uses the Closes-Bug notation
- (10) -> Change-Id
- OB1 :: M3, Made changes to the Sahara project, a cross-projects repository,and explained the changes to participants, then asked participants to write a complete

commit message following the best practices, to describe the changes that were made on the Sahara project. M3: cross"each one of you should discuss with a mentor before submitting the message to the IRC and etherpad doc."

2. Code Quality

Best Practices

Code Quality — Coding Guidelines, Syntax checks and Testing.

M1 gave a brief explanation on hacking style and listed some advantages "Hacking style guide was enforced by reviewers manually, but the process has been automated. Therefore, hacking makes code written by many different authors easier to read by making the style more uniform. (example: unix vs windows newlines)

Call out dangerous patterns and avoid them. (example: shadowing built-in or reserved words)."

M3: gave several use-cases to show how contributors can improve the quality of code by following the Coding Guidelines suggested by OpenStack.

> (M3) : "to ensure high quality code, OpenStack recommends some syntax checks Frameworks such as: (eslint-config-openstack, Hacking, bashate, etc.), and enforces the OpenStack Coding standard. Experience shows that when contributors write code that respect the Coding Guidelines proposed by OpenStack,reviewers spent less effort and smaller amount of time to understand the code, this also, reduces the iterations. Therefore, we encourage you to use the best practices when writing code, commenting on codes, commit messages and testing."

```
An example of an acceptable function docstring is:

def mult(a: Union[intho, float],
         b: Union[int, float]) -> Union[int, float]:
    """Multiple a * b and return the result"""
    return a * b


def some_function(a: FineObject):
    """Do something with a FineObject

    :param: a is used in the context of doing something.
    """
    do_something_with(a)
```

In addition, M9 also said that OpenStack has a large code base, spanning dozens of git trees, with over a thousand developers contributing. As such, common style helps developers understand code in reviews, it also move between projects smoothly, and overall make the code more maintainable.

> M9: "One of the beauties of ESLint is that, despite there being one standardized set of rules created for OpenStack, ESLint permits the overriding of these rules on a **per-project basis** to ensure that no project is hindered by a generalised decision and projects do not have to forgo the use of eslint-config-openstack due to fears of these restrictions. Saying this however, the point of having these standardizations may be slightly defeated if the aim is not to stay as close to the common guidelines as possible."

Furthermore, M9 gave some code samples to the participants that were written and committed by core-contributors at OpenStack. He instructed the participants to explore the code, learn the style in writing code and apply the techniques they have just learned on reviewing code. Then explain their review with their mentors, and commit their reviews to the sandbox project repository.

All the participants were engaged in the activity and after the mentors have gone over their reviews, the participants posted their review on the etherpad doc.

Next, the mentors, showed couple of bad samples of commits that reviewers rejected because the commits violated the code style, which OpenStack enforces. Besides, M9 told participants to avoid such bad practices, which will certainly increase the efforts that they put in to make changes and also increase the time to

3. Insight from lead mentors

OB1 asked M1 and M2 "Can you summarize what the onboarding training is all about, and what values it brings to the OpenStack ecosystem? How has the training material evolved overtime?

M1

```
Onboarding at OpenStack is an intensive program designed for newly
graduated student in mind who are motivated and about to startn their
carrier in open source ecosystem such as OpenStack but lack the technical know-
how.
We give them materials and hands-on training that equipes them to master
the tools, which they will use in making contributions to the codebase;
add new features, fix-bugs, write documentation and participate in
working groups to OpenStack as they join a community of thousands of
developers from hundreds of companies worldwide.

Students will also learn how to use a prepared development environment
to test, prepare and upload new code snippets or documentation for review.

This year, Lenovo is sponsoring the onboarding event. But the Onboarding
is organized and run by people embedded in the community, the fast-track
course gives students a more accurate taste of what working in the
community is really like and the opportunity to ask experienced
contributors questions and gain more insight into their work with OpenStack.

There’s help beyond the classroom, too: attendees can join an ongoing
 mentoring program.

As the training evolves, our focus continually shifts towards providing a
highly interactive course where students can learn about the social norms,
modes of communication and variety of possible contributions through
 experience as opposed to lectures.
```

M2 supported M1 response and added that:

M2

```
"Onboarding new contributors is incredibly important in any open source
ecosystem and particularly one as large as OpenStack.There’s a constant
flow of people joining our community and some moving on to other endeavors.

The best way to maintain a healthy community is to educate newcomers and
give them the tools they need to become effective contributors.
One of ways OpenStack does this is through the two-day long Upstream
Institute Training offered prior to each OpenStack Summit."
```

4. Project Onboarding

   Towards the end of the two-days event,

   - M1 said: "During mentoring, OUI participants are strongly encouraged (**required**) to

   join at least one project team. Once signed up to join a project team, they are assigned mentors to follow them up."

   - M2: "Project Onboarding gives participants a chance to meet some of the project team and get to know the project. Participants will learn about the project itself, the code structure/ overall architecture, etc, and places where contribution is needed. Participants will also get to know some of the core contributors and other established community members. Ideally, participants will know/ have completed the OUI basics."

5. Mentoring Program

   Prospective Participants
   All 72 participants signed up to participate for

   the mentoring program.

   P1, P2, P3, …, P72. Moreover, they indicated interest in different project teams.