**Self-Intro**

* Good Afternoon / Thank you for giving me this opportunity to introduce myself / my name is Faizul Ahamed /I'm from erode tamilnadu.
* My schooling and diploma were completed in my hometown / I studied engineering at crescent engineering college in Chennai, Tamil Nadu.
* I like playing cricket, long drives in the car and bike, and spending time in hill stations and peaceful areas.

**Miracle Systems**

* From 2005 to 2011, I worked for Miracle Systems as a system engineer.
* I will be responsible for installing operating systems, installing and troubleshooting hardware.
* Maintaining inventory, coordinating with the vendor and the customer, and ordering product from the vendor based on the requirements of the customer.

**HCL**

* Later, I got an opportunity with HCL info systems in Bangalore as an associate engineer.
* My roles and responsibilities are maintaining python-based applications. Maintain up to date code documentation, participating in code reviews and Contributing in all phases of the development lifecycle

**IMSI**

* Then I got a job at IMSI India Pvt Ltd as a Devops support engineer.
* There my roles and responsibilities are I will co-ordinate with developer and understand the concepts of applications, which GIT workflow they are using (5 different Work flow is there, 1. Basic workflow, 2. Feature branch workflow, 3. Git Flow, 4. Gitlab flow and 5. forking workflow)
* Deployed either a testing server or a UAT server .
* Also I co-ordinate with operation team and I will prepare the infrastructure to run the application.
* We used Bitbucket code repository
* **Precision**
* My last company is precision tech serve private limited
* There my roles and responsibilities are same like I handled in the previous project.
* I will coordinate with all other teams to obtain the latest information regarding current processes or any important organization's work.
* I am responsible for making sure everything runs smoothly on my shift as shift leader

**My day to day activities are:**

* **Developer Push Code**🡪**Developer Mail Conformation** 🡪**Team City (Check Builds Successfully, Note Down Logs)** 🡪**Jfrog** **Artifactory (File Created & Stored Here)** 🡪 **IBM UCD(Urban Code Deploy) (One-Click Deployment / Testing Server or UAT Server)**
* **Snapshot Version**🡪**We will deploy in testing server**
* **Release Version**🡪**After all testing over**🡪**Deploy to UAT server or Production Server**

1. I worked with the deployment team (Architect will implement continuous integration (pipeline)
2. Whenever a developer pushes code from user branch to development branch or merges to development branch, I will check the build & build version with team city by using the developer mail confirmation(Like Branch name, version number, some other message). In the logs, I will note down all the details. (In team city also snapshot version, release version folder will be there)
3. Then I will log into artifactory, where there are project paths and snapshot versions. Here I will cross check file was created in artifactory with the correct version (by using the team city build logs)
4. If it's snapshot version then we will deploy to testing server.
5. After logging into UCD tool, I will find all the testing servers IP addresses there. After clicking on the particular version, it will get deployed (i.e., one-click deployment).
6. In UCD we will check the deployment logs (failure or success)
7. If all the testing get over then they will merge from development branch to master branch.
8. Again we will get a mail confirmation from development team. All the steps will be the same, except that we will build in release version and deploy to UAT server.
9. We used only python

Deploy folder :

In server we have the deploy folder

UCD can do code, put into the deploy folder.

I worked with On-premises server.

Monolithic services / Server based deployment /

Microservices / Docker🡪 Kubernetes

Agent Failure (Build Issue) (Code

User name: myfaizul

Email: [myfaizul58@gmail.com](mailto:myfaizul58@gmail.com)

Jira : myfaizul.atlassian.net

I’m ready to work with anykind of DevOps technologies like Git, Jenkins, Maven, Kubernetes, Dockers, Docker Swarm, or EKS

Even if I don’t have an hands on experience I will manage and I will be expert very soon bcos i like devops

And practically talking if you take me for your projects I believe I will get to work on some of technologies and not all. So I’m confident for even if I don’t have hands on experience on some devops tool I will easily understand the concept and I will give my 100% real and honesty effort. That’s my promise.

Hopefully, I answered the majority of your questions and I hope you are satisfied to take on this project as well

I look forward to your feedback and am eager to expose my talent as well as learn some new tools and technologies

Loan IQ:

Loan IQ essentially provides a single data model that aspires to create a global platform. [Automation](https://www.hcltech.com/autonomics-and-orchestration), based on the vital data, helps reduce errors and operational cost. Among the sustainable benefits of a loan IQ model are seamless functioning of the entire loan lifecycle and greater control over profits.

Openshift:

**OpenShift provides a common platform for enterprise units to host their applications on cloud without worrying about the underlying operating system.**

OpenShift is a**family of containerization software products developed by Red Hat**. Its flagship product is the OpenShift Container Platform — a hybrid cloud platform as a service built around Linux containers orchestrated and managed by Kubernetes on a foundation of Red Hat Enterprise Linux.

Control-M:

**Control**-M's inbuilt run time parameters are Order ID, Start Time, End Time etc. CD Process ID: A process ID is created within Connect Direct when a process is submitted. This number is used to retrieve the logs, status, statistics of that particular process.

Control-M simplifies application workflow orchestration. It makes it easy to define, schedule, manage and monitor workflows, ensuring visibility and reliability, and improving SLAs.

Bamboo

Bamboo is a**continuous integration server** offered by Atlassian. Bamboo allows you to automate the release management of a software application, creating a continuous delivery. This CI/CD tool enables your DevOps team to spend less time creating and running time-intensive yet redundant scripts and more time building the code that matters.

**Git & Github**

Git is the free and open source distributed version control system that's responsible for everything GitHub related that happens locally on your computer Ex: We have an app then later we added new features in app / After update new features app is break or corrupted / So here it will capture all the history details (Like when it was updated, who updated, etc..) whenever we changed or updated so we can easily go back to previous version and make it app run smoothly.

Github is the most popular one and lot of new features / Github is the platform online website that allows to host Git Repositories. Repository is just a folder where all the changes are saved / Git repository will collect all the history of projects and the folder name is .git

|  |  |
| --- | --- |
| **GIT** | **GITHUB** |
| **GIT** is one of the best version control available now | **GitHub**: Developers need a web/cloud-based code hosting platform |
| It is so flexible. You can create your own workflow strategies using git | Useful for version control |
| Git is the tool | Enables effective collaboration |
| A tool to manage our source code history | Download projects and file in one go |
| Installed & Maintained in your local system | Easy evaluation of each other's wor**k** |

**Git Commands**

* Push - pushing sends the recent commit history from your local repository up to GitHub.
* Pull - a pull grabs any changes from the GitHub repository and merges them into your local repository.

**Command is🡪 git reset commit-ID or git reset --soft commit-ID** (It will delete after this commit what & all happened) / **git revert commit-ID or git revert --soft commit-ID** (it will delete only one commit)

Most useful Command: **sudo apt-get install git –y /** **git init / git –version /git status / git log /**

**git config --global user.name "user name" / git config --global user.email** [**youremail@domain.com**](mailto:youremail@domain.com) **/**

**git config –list**

**/ git config user.name / git config user.email /**

**Clone from server (Github) to local**

**git clone repository-url address**

**Clone from local to server**

**git init** (In any location, Same repo name create in github) **/ git add . or particular file name /**

**git commit –m “any message” / git branch –M master** (Check branch name either master,main or other) **/**

**git remote add origin repo-url address / git push -u origin master or git push –f origin**

**git merge branchname** (It will create a new merge commit and existing branches are not changed in any way)

**git rebase branch1(from) branch2(to**) (If we want to moving or combining a sequence of commit one branch to another / alternate for merging) (merge is like a copy) (it will re-writes the project history)

**git cherry-pick commitID** (It will pick only one specific commit from another branch) (1st we need to go base branch)

**(squash) git rebase -i commit id (**Multiple commits into single commit) (We are changing the GIT history) (Once squash you can’t delete any particular commit) (b4 squash think very well)

**or git pull --rebase origin main / git push or git push origin main**

**git stash** (We will send to backstage or we can say hide in the background) **/** before complete the task if any other emergency task need to commit. Then we can use stash

**git stash pop** (come to stage again) **/ or git stash pop 0(stashnumber)**

**git stash list**

**git stash clear** (Delete the stash)

**git remote –v** (URLs that Git has stored)

**tree** /

**git branch branch\_name** (Branch Creation)

**git branch** (Current Branch)

**git branch –av** (To list all branch)

**git branch -D branch\_name** (Branch Deletion Locally)

**git push origin --delete branch\_name** (Branch Deletion to server)

**git checkout branch\_name** (Branch Switch)

**git push origin branchname –f** (Force Push)

**git branch New\_branch\_name1 branch\_name2** (create branch without checkout branch) /

**git checkout -b new\_branch\_name** (while checkout create new branch)

**git push origin branchname** (to merge)

**Difference b/w git fetch and git pull?**

|  |  |
| --- | --- |
| **Git Fetch** | **Git Pull** |
| **Git fetch** brings the changes from the remote repository to local repo then later we will merge manually | **Git pull** brings the changes from the remote repository to local repo and it will merge automatically (git pull = git fetch + git merge) |
| Git fetch no chance of merge conflicts | Here merge conflicts may occur |
| Git fetch will update in remote not in local so we need to merge to update in local workspace | It will update both remote and local branch / local branch need to be committed before running git pull |

**What is Git log? What information in git log help you?**

* Git Log is helpful command that shows us the history of over repository, information like the commits, the branch information, The merge, the fix, etc..
* This is critical for us in many cases that we want to revert our changes or we want to cherry pick some changes and test those changes in our branch

Confluence:

Confluence is a team workspace where knowledge and collaboration meet. Dynamic pages give your team a place to create, capture, and collaborate on any project or idea.

To avoid any miscommunication and promote a good and clear understanding among all people working in different stages of the SDLC, a collaboration tool is needed which can serve the**purpose of providing a** platform**for all to get connected,** share information, get updates, communicate with each other, etc.

**Agile**

* Agile is a popular methodology that offers guidance to teams on how to approach their projects
* Agile is based on the values and principles outlined in the agile manifesto which was published on 2001.

**Agile have 4 factors:**

1. Individuals and Interactions
2. Deliverables and documentation
3. Customer collaboration
4. Quick response to change

* Agile teams break their projects down into short phases called iterations. They make continuous improvements and provide deliverables within each iteration and provide high quality deliverables on a regular basis

**Kanban:**

* Kanban is a project management framework that boosts workflow visibility. Teams use a Kanban board to categorize their tasks by status.
* In a Kanban project teams use card to symbolize their tasks on the Kanban board.
* Cards are moved to a different column when the task status changes. For example: Team member might move a task from a column marked to do to a column marked doing or new to doing

**Kanban Principles:**

Kanban teams followed 4 main principles

1. Start with the most urgent task
2. Make small changes regularly
3. Maintain existing roles
4. Encourage Leadership (Encourage every team member to act as leader)

**Agile and Kanban Similarity:**

* Kanban falls under the agile umbrella meaning the two are linked in project management. They represent an alternative to traditional project management which includes framework such as waterfall agile and Kanban have a few features in common.

1. BITE SIZE TASKS

* The 1st similarity is that both Kanban and agile break projects into smaller chunks to accelerate work process and deliver result on time. So it’s easier to manage.

1. TRANSPARENCY

* 2nd similarity is Kanban and Agile teams have a clear understanding of their tasks and provide regular updates.
* Kanban boards clear view of what tasks are planned in progress and completed / Agile team communicate openly and regularly so every team is kept in the loop

**Difference between Agile and Kanban**

|  |  |
| --- | --- |
| Agile | Kanaban |
| Agile is a overarching methodology that offers general guidelines to varying teams. | Kanban is a visual framework that provides specific details on how to manage projects |
| TIME: Agile teams work in short, time boxed periods | Kanban projects have no set time structure |
| TASKS: Agile teams use metrics known as story points to eliminate the problem for more accurate planning. | Kanban tasks vary in length and effort, making it difficult to plan resources. |
| VISUALIZATION: Agile teams may use boards or other visual aids but they are not mandatory | Kanban teams use a Kanban board to visualize workflow |
| MEETINGS: While daily stand-up meeting are a key requirement in Agile projects (Where team discuss past, future plans) | Kanban daily meetings are optional and team member might instead choose to communicate status update via the Kanban board |
| TEAMS: Agile projects have cross-functional teams, This is the key element of the Agile | The Kanban structure is looser meaning the framework is also suitable for functional team that operates separately |

**Scrum Vs Waterfall:**

**Waterfall:**

Typically goes through a lengthy planning process, it will take several months followed by building the products which again take many months and then testing the products. Reviewing and deploying the products.

Several problems with this method first of all the planning must be completed before any work begins and in most cases the planning is done without entirely understanding the project

**SCRUM:**

Scrum an implementation of Agile the process is broken up into smaller pieces.

1st we do just enough planning to get started with building the minimal features set.

We build what was planned.

We test and review that small features set and get it ready to ship

Then we will review and ready the products.

It will repeatedly for every time.

Several incremental releases called Sprints. Sprints usually takes from one to three weeks and you just keep repeating these sprints until your product feature complete. May be you end up shipping your products in Sprint 2, Sprint 3, or even further. But you eventually end up with a shipping product

**In SCRUM there are three key roles that are needed for the framework to work well.**

* 1. Product Owner: Product owner has the bright ideas turn into products
  2. SCRUM Master: Is a servant leader to the team responsible for protecting the team and the process running the meetings and keeping things going.
  3. TEAM: The team can be made up of developer’s testers, writers and anyone else that helps in building a product. Team members often play multiple roles, somedays developers may end up doing tests or testers may end up writing either way the team works to get the product done

**There are 3 artifacts or documents that are used in SCRUM**

1. Product backlog: The product owners create a prioritized list of features known as user stories that could go into the products. This list evolves and changes priority with every sprint
2. Sprint Backlog: Highest priority user stories go into the sprint backlog and are committed to the next sprint
3. Burndown Chart: Show the progress during a sprint on the completion of tasks in the sprint backlog. This chart should approach zero points as the work is being completed

**Three Ceremonies: 1. Sprint Planning 2. Daily SCRUM 3. Sprint review**

**SCRUM WORKFLOW:**

Product Backlog🡪Sprint Planning🡪Sprint Backlog🡪Sprint (1 to 3 weeks and daily scrum)🡪Potentially Shippable Products🡪Sprint Review🡪Retrospective (Repeat this workflow for each sprint)

**Software to manage the scrum workflow:**

**Utility** **software** has been built around the scrum process to help people filling the three scrum roles manage the 3 artifacts and better run the three ceremonies

**BitBucket**

* It’s useful to store the developer code
* Bitbucket is a Git-based source code repository
* Bitbucket offers both commercial plans and free accounts with an unlimited number of private repositories.
* Bitbucket is a system for hosting version control repositories
* Bitbucket makes tools that use Git
* Bitbucket is written in Python

**BitBucket Features**

* Bitbucket integrates very well with JIRA, originally built as a bug tracker, but is so flexible now it can be used for: bug tracking, issue tracking, service desks and project management.
* Pull requests and code reviews
* Unlimited private repos
* Branch comparison and commit history
* Bitbucket Mac and Windows client called SourceTree; Android app called BitBeaker
* Bitbucket for Enterprises, called Stash
* Integration with tools like Jira, Crucible, Bamboo, Jenkins, HipChat
* Built-in continuous delivery, issue tracking and wikis.
* Bitbucket Snippets allow you to create and manage multi-file snippets of all kinds.
* Third-party integrations allow you to integrate Bitbucket into several facets of your existing workflow, making the complete development process much more efficient.
* Bitbucket supports Git Large File Storage (LFS) which means shorter clone and fetch times for those working with large files.

**BitBucket Vs Github**

* most basic difference between GitHub and Bitbucket, it is that GitHub is focused around public code and Bitbucket is for private.
* GitHub has a huge open-source community and Bitbucket tends to have mostly enterprise and business users.

**Team City**

* Team City is a tool developed by JetBrains to manage different builds of projects and perform continuous integration and continuous deployment.
* It’s not open source it provides a free version with limited abilities
* When we installed team city its comes pre-bundled with the most widely used dependencies (Like maven, ant, git etc.. that you required)

How It’s Work:

When developer pushes code🡪Server🡪TeamCity🡪Deploys if builds are successful

Every build even if it’s successful or not, comprehensive report is created which is then provided to the developer and the admin.

If any build errors or lot of bugs then teamcity automatically identifies those who are responsible

It’s also allow install extra plugins that are not provided with the teamcity, Install it externally and then use it as a plugin inside teamcity so it allows for extensive configuration

Team City Installation:

Install Centos in AWS🡪

TeamCity is used **to build and test software products in an automated manner**. It provides rapid feedback on every code change, reduces code integration problems, and leads to more effective teamwork. Many popular games, websites, banking systems, and all of JetBrains products are built with TeamCity.

**TeamCity from JetBrains is a popular continuous integration server that supports a variety of different version control systems and build runners**. Octopus Deploy and TeamCity can work together to make automated, continuous delivery easy.

TeamCity builds the code and runs tests, while Octopus takes care of: Distributing applications to all the remote machines, securely. Environment-specific configuration, like connection strings. Configuring IIS sites and installing Windows Services.

**TeamCity is a CI server written in Java**. It allows developers to integrate, code, and is easier to configure with simple steps

Octopus Deploy is an automated deployment and release management server. It is designed **to simplify deployment of ASP.NET applications Windows Services and databases**.

Jenkins is an open source continuous integration tool, while TeamCity is a proprietary offering from JetBrains. **TeamCity is easier to configure and more straightforward to use, while Jenkins has a rich plugin ecosystem and integrations**.

**004 Team City Project:**

A project in TeamCity is **a collection of build configurations**. A TeamCity project can correspond to a software project, a specific version/release of a project or any other logical group of the build configurations. The project has a name, an ID, and an optional description

**005 Create Pipeline:**

Import Projects🡪Configure Snapshot Dependency🡪Configure Artifact Dependency🡪Run Simple Chain🡪Configure Trigger and Checkout Rules🡪Add VCS Trigger🡪Restrict Checkout Scope🡪Complete Chain with Tests🡪Takeaway

Go to **Administration | Projects** and click **Create project**.

Open the **Dependencies** settings tab (you might need to click **Show more** to display this item) and click **Add new snapshot dependency**.

Open the **Dependencies** settings tab and click **Add new artifact dependency**.

TodoImage build with the **Run** button

Open the **Triggers** page and click **Add new trigger**.

Opposite our only VCS root, click **Edit checkout rules**.

A build chain is a sequence of builds connected with snapshot dependencies. A snapshot corresponds to a certain commit in the source code.

**Jfrog Artifactory**

Artifactory is a repository manager created by [JFrog](https://www.jfrog.com/). A repository manager is a dedicated server application designed to manage binary components for the application that we build.

Improve build stability since you are less dependent on external repositories.

**UCD**

UCD uses a distributed server agent model, It will communicate with the agents on your target systems. These agents are responsible for automating the deployment processes.

UCD can do code put into the deploy folder.

**Jira Software**

* Jira Software is a software development tool that gives teams the power to release great software early and often
* Plan the big picture, track the details, and keep everyone in the loop.
* Using customizable workflows and best-in-class integrations, teams from startup to enterprise can use Jira Software to ship with confidence and speed.
* Select the Atlassian switcher to switch between products.
* Jira connects to the tools you use everyday making it easier for you to get more done. (Guthub, Google Sheets, Gitlab, Jenkins, etc..)
* D

Tools Details: Login🡪Create a project🡪Planning (🡪Roadmap🡪 Create Epic(What needs to be done) 🡪Backlog 🡪Board 🡪 Assign 🡪Export🡪

**Sprint**

* A sprint is a time-boxed period, with a specific goal and outcome. Allocate work that aligns with your goal and can be achieved on time.
* Plan our Sprint: Drag Issues from the backlog section, or create new issues, to plan the work for this sprint, Select start sprint when you’re ready

While Creating Repo🡪We need to choose language (Python, C, Java, etcc..) So it will create language based repo