Infrastructure Design

1. Introduction

# Given a simple application with UI (React/Angular, etc.), Rest APIs (Spring Boot/etc.), Cache (Redis/etc.), DB (RDBMS). define the deployment based on the following constrains (Assume the Cloud environment is AWS or of your choice if so, do mention the environment)’.

1. Deployment Solution

# I tried keeping all information as mentioned in problem statements. Taking two VMs to deploy UI and App component to make it load balanced. Below example is very high level of flow from App to the services and how request will be delegated to hosted zone.

# My App is the application (product UI) which will be used by the end user, once application sends request to any service then it will be via internet and accessible through https protocol. Initially request will be delegated to the DMZ zone and then it will be delegated to trusted zone and all services will be deployed in trusted zone.

# 

# As illustrated in below architecture diagram, we can divide our services based on group like platform service (basic service which must be executed) and business services (business micro services) and UI components. Below example represents primary environment and secondary one will be replica of primary. **Component Group-1** – All basic services related to platform/framework (this could be company specific services as well) **Component Group-2** – All business services related to business functionalities **Component Group-3** – UI Services, this could be app or any API specific documentation (optional) Similarly another VM also will be loaded as primary and these two VMs will be loaded balance though any hardware load balancer like F5 LB.

Shape

Description automatically generated

A picture containing timeline

Description automatically generated

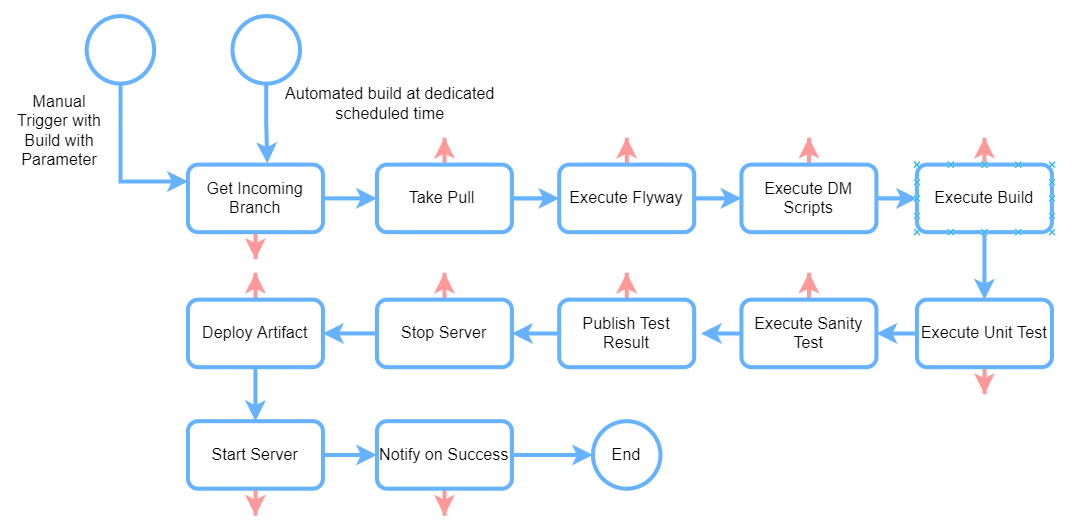
# Above architecture diagram is high level flow of API/services from the App, as illustrated any service call will be first generated from the app and app could be accessed via a browser and since API is access over https then complete content will be encrypted and send. At premises, https will be loaded as soon it reached to the load balancer. App is going to call an API with load balancer (could be F5 or any other). This LB will have configuration for API gateway and Juul can be implementation of API gateway. Jull will have all configurations and service URL which is deployed in Trusted zone.

# Direct service URL will not be exposed to the client, means App cannot call direct service URL instead it will be through load balancer and then will be delegated via gateway.

# As mentioned in above diagram, once requested reaches to Jull, it will identify SD and through SB request will be delegated to the product services and these services are deployed in two difference VMs

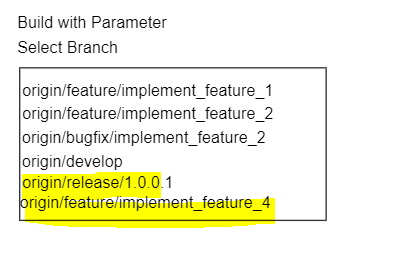
1. CICD Solution

# CICD stands for continuous integration and continuous delivery/deployment. This is nothing but automated process for getting code build from centralized server, verify testing, deploy in server, execute automation test, and notify users. This can be extended to deploy artifact in production environment. Below illustration is high level steps/process CICD take to deploy artifacts in production environment.



As per the problem statement, I tried keeping above diagram in very simple way. Once CICD pipeline is setup first point is to get code from centralized repository (it could be GIT) and take a pull for given branch. Depending on requirement, we my need to execute few DDL/DML, to do that we can setup Flyway job as next process/job which can take care of DDL and DML execution, once it’s done then build process will start. Once build is success there will be test suits which must be executed and if test results are within threshold (we can set the same), we can publish the test result and start deployment. To do this, first Jenkin to execute script to stop server, deploy artifact in server and start the same. Since we have clustered environment, deployment can be in rolling process i.e., stop one server deploy the same and start the server and once server is up then go to another server to do the same action. If any issues during any steps then process will be rollbacked and notification will be sent to the DevOps team. If any DDL/DML or Flyway is executed, then flyway should execute prior version of flyway script so that DDL and DML can be rollbacked.

# In Jenkin we can setup dedicated build based on environment like Dev/QA/UAT/PROD. If its dev environment scheduler Job will pick develop branch and proceed with the pipeline has been setup and similar for another environment as well. We can have another option where input branch will be passed by the user who will trigger the build, in this case use should be able to select branch from the repository like, if for QA team there will be an option to deploy any branch in QA environment, it could be develop/feature/release/bugfix depending on requirement and urgency. Belo snippet is just for a reference how user can select branch in Jenkin and trigger the build.



1. Git Solution

# I am proposing solution as below, there will be 4 type of branching strategies we can have. Based on situation we can create feature/release/hotfix as illustrated in below diagram. Chart, line chart Description automatically generated

# **Master Branch** Maser branch will be main branch which will be deployed in production and write access to this branch should not be given to any one in the organization as best practice. This will be updated by automated tool like Jenkin etc.

# **Hotfix Branch** Hotfix branch will be created only if there is emergency fix required in production and this branch must be created from master branch. We can use tool like Bitbucket which can create Hotfix branch from the master (this can be created manually but always it will be good to use tool to maintain version and avoid any issues due to manually activity). Once Hotfix is done, this will be merged to master and then there will be a pull request from master to develop to make hot fix in sync with develop branch as well.

# **Release Branch** Release branch is the branch which should be created based on regular release, and this branch will be created from develop branch. This release branch should have feature available which is planned in the release and at last will be deployed in QA and upon verification if everything is fine then it will be deployed in UAT/Stage/PROD. Once it’s deployed in PROD, release branch changes will be merged to master and there will be pull request from master to develop to make all changes in Sync (there will be a version change in every production promote and version must be updated in the develop branch for further releases).

# **Develop Branch** Develop branch is main branch which will have feature ready by the developer and once Unit testing is done there will be a pull request from feature to develop branch to merge all changes in the develop. A developer should not have read access to this branch; however, Lead/Architect can have read/write access to this branch.

# **Feature Branch** Feature branch is the branch which must be created from develop branch (sometime from release branch based on architecture decision and business requirement). Each feature branch should be created for a feature or functionality planned in the sprint and once changes are done then, developer should create pull request from feature to develop and assign senior engineer/lead/architect for code review.

# **Bugfix Branch** Feature branch is the branch which must be created only to fix bug

# **Code Review Process:**

# As mentioned in Feature branch details, once develop is done with the implementation and all UT is fine, then developer should create pull request (PR) from the base branch and since feature branch is created from develop then there will be pull request from feature to develop branch and develop and keep senior engineer/ lead and architect as reviewer. During review if any issues found in the code, PR should be declined with proper comments and comments can be given in each file even in each line if its required. Once PR is declined, develop should be notified and modify code as per the declined PR comments and once all changes are done, develop should create PR again and keep all required people for review. Once changes looks fine, Lead/Architect can approve PR and finally marge the same PR to the develop branch.

1. Assumption

# As explained in my interview process, I have mentioned that I have developed many products in Airline Cargo Industry. These products are deployed in premise whereas its is being access via cloud services. I didn’t get an opportunity to work on cloud which I am looking for. However, I believe that deployment concept will be same in cloud infrastructure as well.