**CI/CD Documentation**

**For CI Pipeline of VAM Reference microservice**

**Assumptions:**

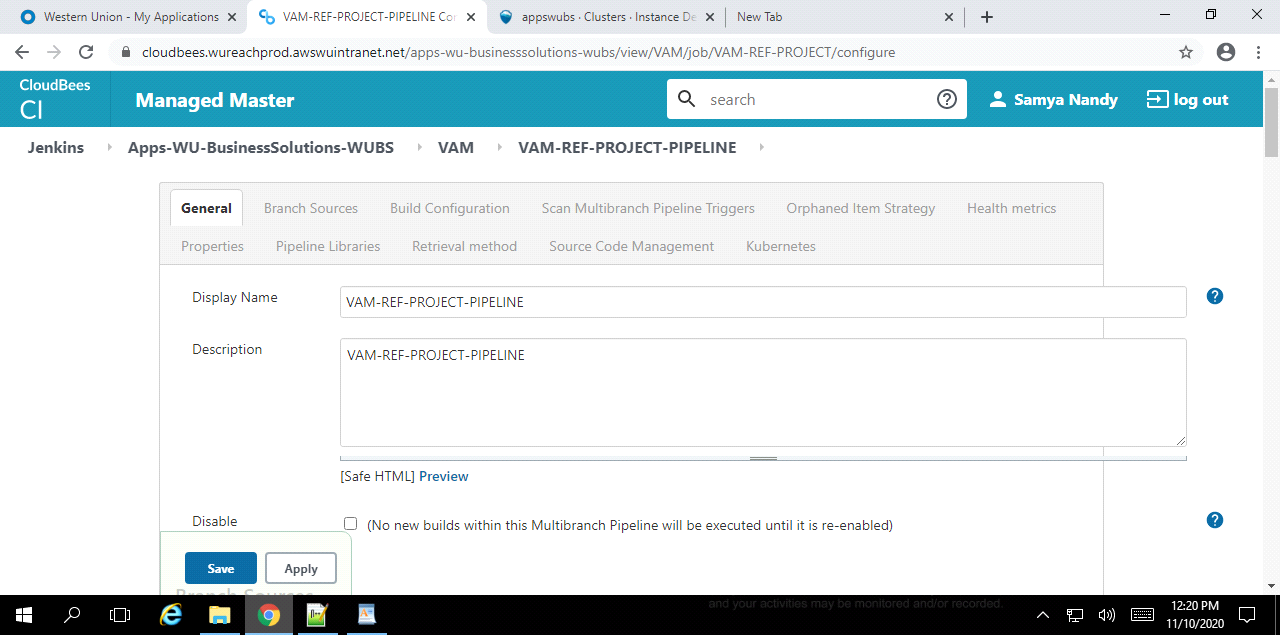
Need to access to CloudBees Jenkins.

**Prerequisites:**

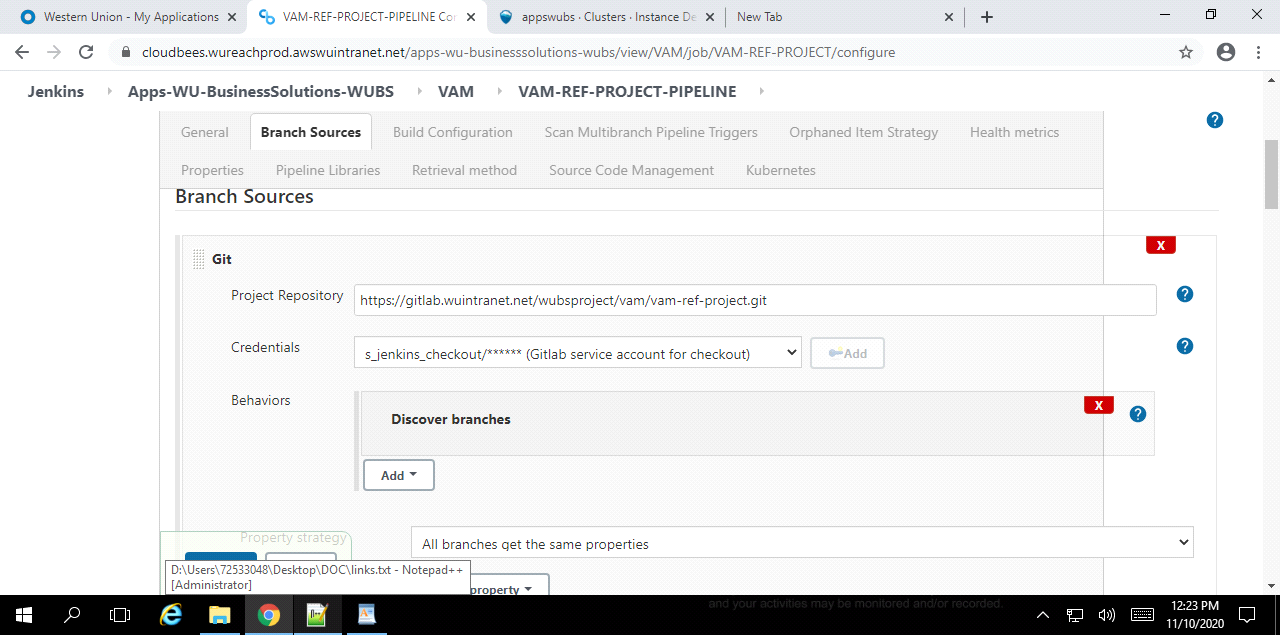
Application BU specific folder to be created by the CCOE team in CloudBees Jenkins.

To configure CI Pipeline of VAM reference microservice we need to follow the following steps:

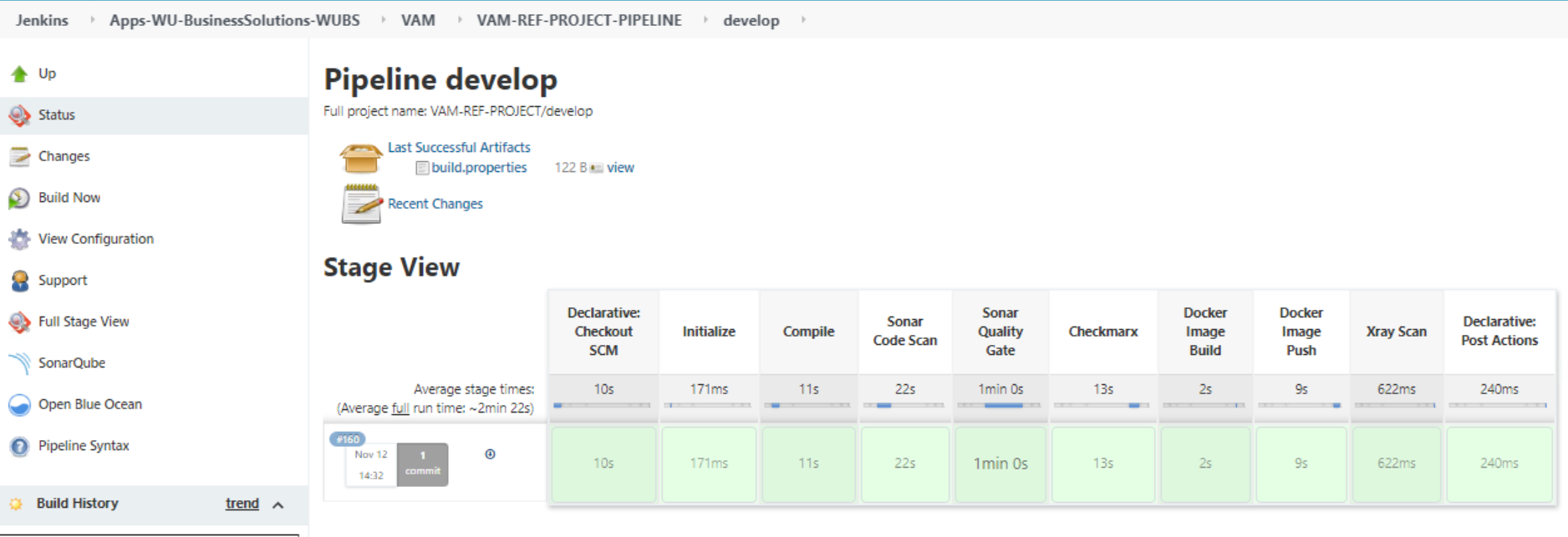
* Create multi branch pipeline



* Integrate gitLab with multi branch pipeline



Below the CI job status-



Capture the stage details in the Jenkinsfile mentioned in the gitlab repository-

<https://gitlab.wuintranet.net/wubsproject/vam/vam-ref-project/blob/develop/Jenkinsfile>

**Stage Details**

* **Initialize Stage**

In this stage artifactory details are initialized, image version captured and email recipients to be added.

Below are the jfrog artifactory and the repository details-

JFrog Artifactory - <https://artifactory.wureachprod.awswuintranet.net/artifactory/>

Repository - wu-businesssolutions-wubs-docker-dev-local

Artifactory URL - <https://artifactory.wureachprod.awswuintranet.net/artifactory/wu-businesssolutions-wubs-docker-dev-local/vam/vam-ref-project/0.0.1-SNAPSHOT>

* **Compilation Stage**

In this stage maven artifacts to be compiled.

* **Sonar Code Scan**

In this stage Sonar Code Scan to be performed in the compiled code.

* **Sonar Quality Gate**

In this stage Sonar Gate checks will be performed with the compiled code. If the quality Gate fails, then the build will fail with the quality gate failure.

* **Checkmarx Stage**

In this stage Checkmarx will be performed with the compile code.

* **Docker Image build**

In this stage the image of the VAM reference microservice to be built with specific version via dockerfile.

* **Docker Image push**

In this stage the specific docker image will be tagged and pushed in specific jfrog docker repository.

* **Xray Scan**

In this stage Xray Scan of the pushed images (as part of build artifacts) to be performed. If vulnerability found then, the CI will be unstable.

* **Post Action Stage**

In this stage if the CI build is successful then notification emails to be sent to the intended recipients.

**For CD Pipeline of VAM Reference microservice**

**Assumptions:**

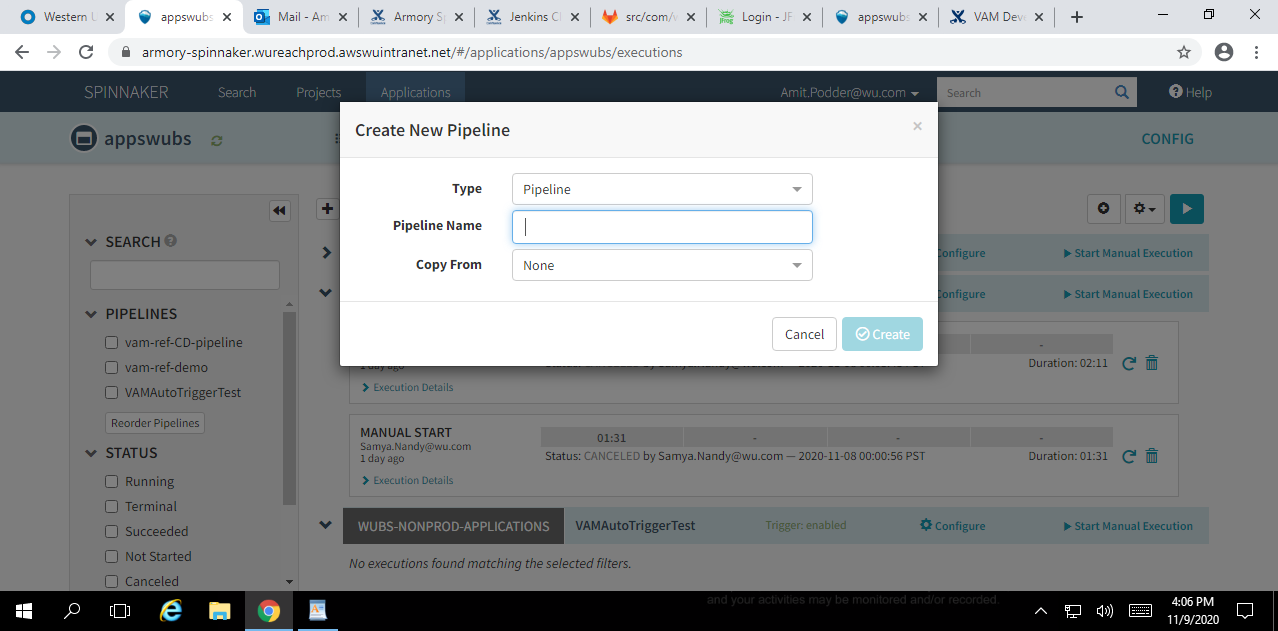
* Target K8s deployment environment is configured in Spinnaker.
* Spinnaker is enabled to fetch artifacts from Gitlab account.
* We will use Gitlab repository as Helm Chart store. The chart package.tar.gz is committed into the Git repo. The repo also contains the chart’s override value.yaml files for different environments.

**Prerequisite:**

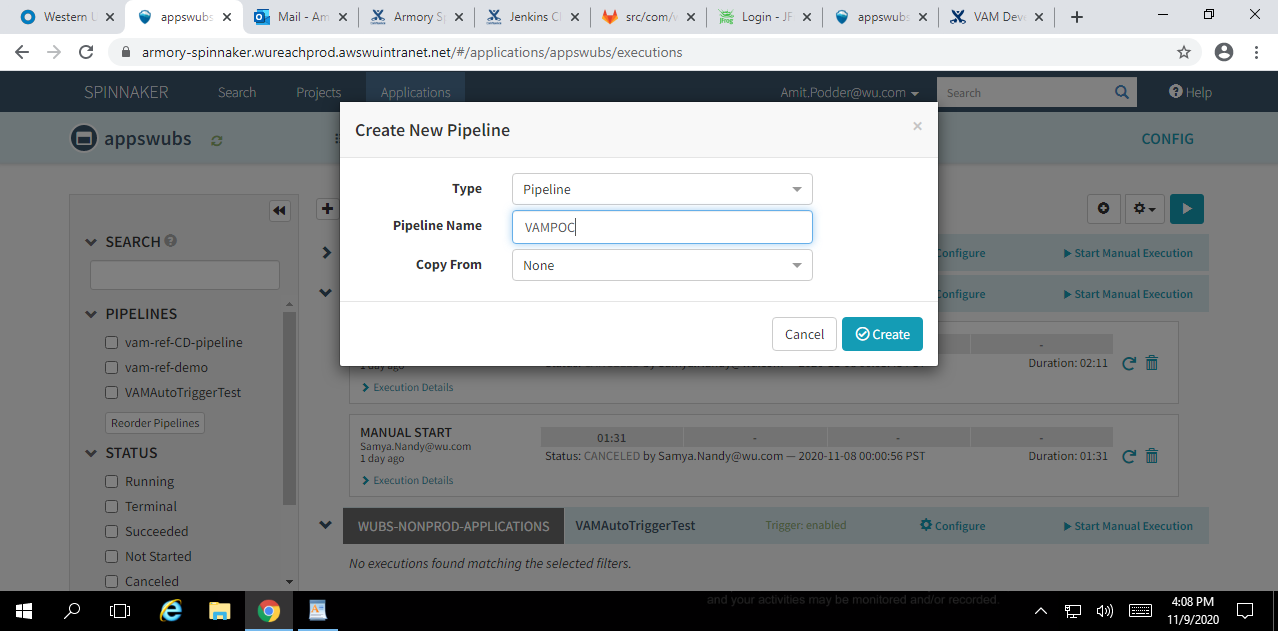
* Spinnaker Application creation for a particular project.
* Give read/write/execute permission to developers to create pipeline on that application.
* Created EKS cluster in AWS.
* Created Account of the EKS Cluster in Spinnaker for Deploy stage.
* Created GitLab account in spinnaker to get Helm packages in bake stage.
* Created controller to get CI Jenkins jobs in spinnaker.
* Created secrets in K8s cluster to pull the docker images from jfrog artifactory.

Sample pipeline created in Spinnaker for VAM reference microservice for CD. Steps are as follows-

* **Create new pipeline:** Under the assigned application "appswubs" we created a new pipeline. A dialog box appears on the screen, we need to put the pipeline name in that.



Once we enter a suitable name, click on create button.

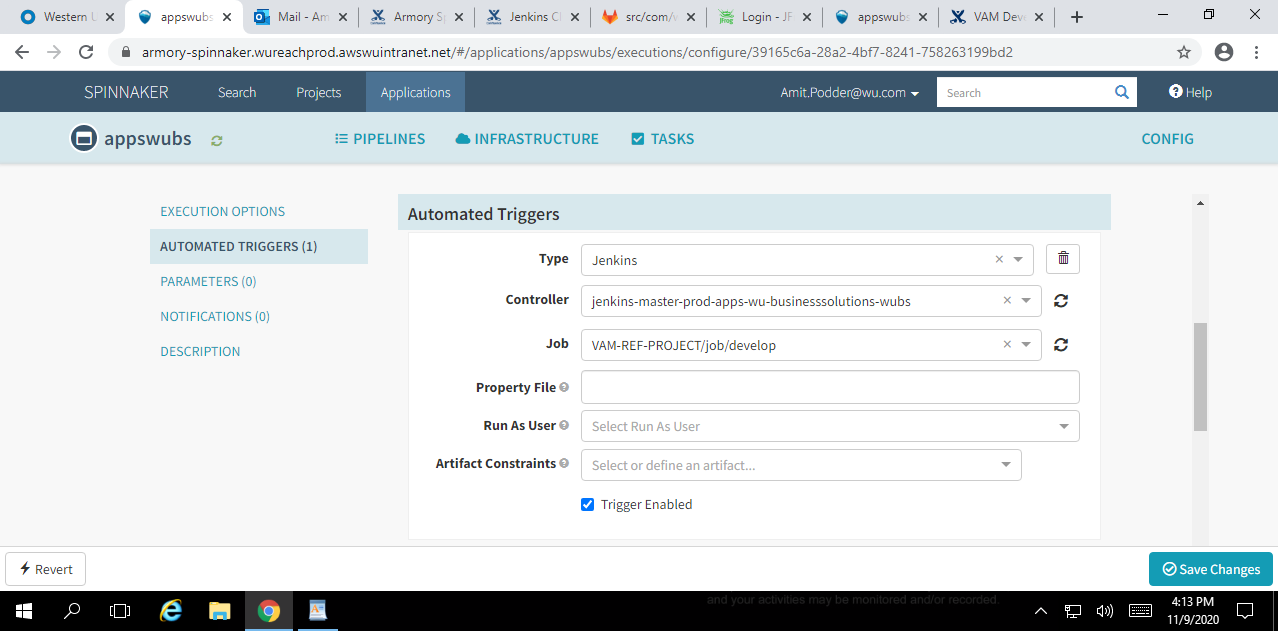


* **Configuration Stage:** Once our pipeline is created, we add the Automated Trigger based on which our CD will be triggered. There are various options for triggering. We are using Jenkins for our CD pipeline.

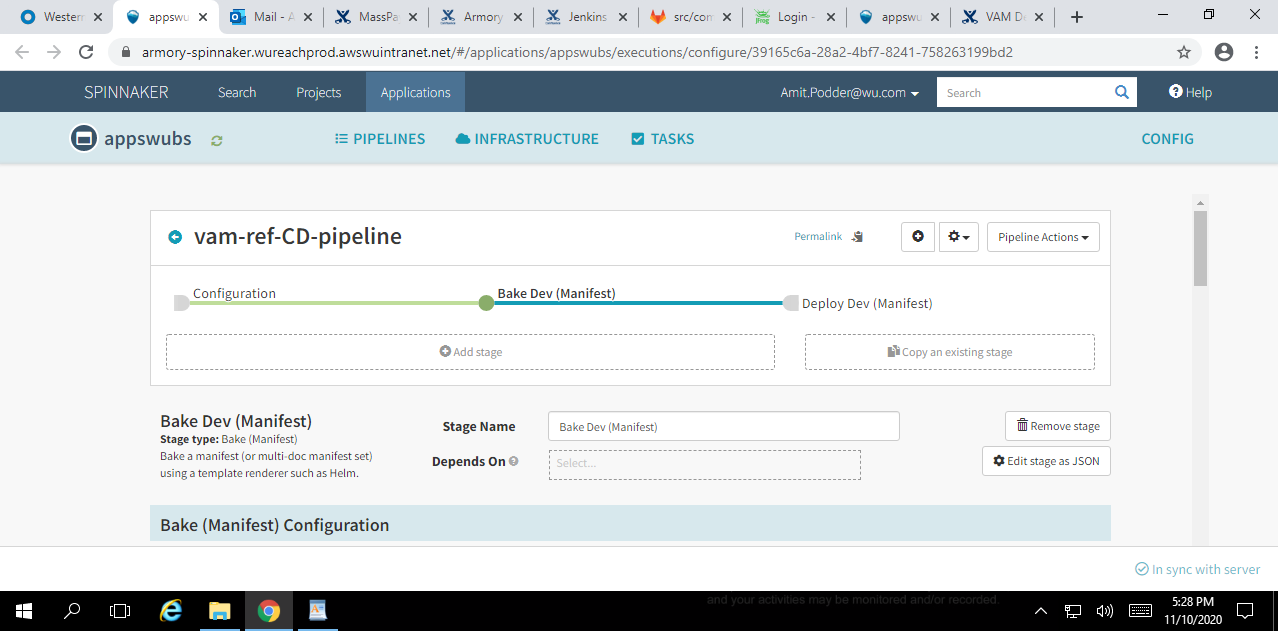
Controller: This is Jenkins master for your BU. Select the appropriate controller for our application

Job: Select the Job that we need to run from spinnaker to get the image.

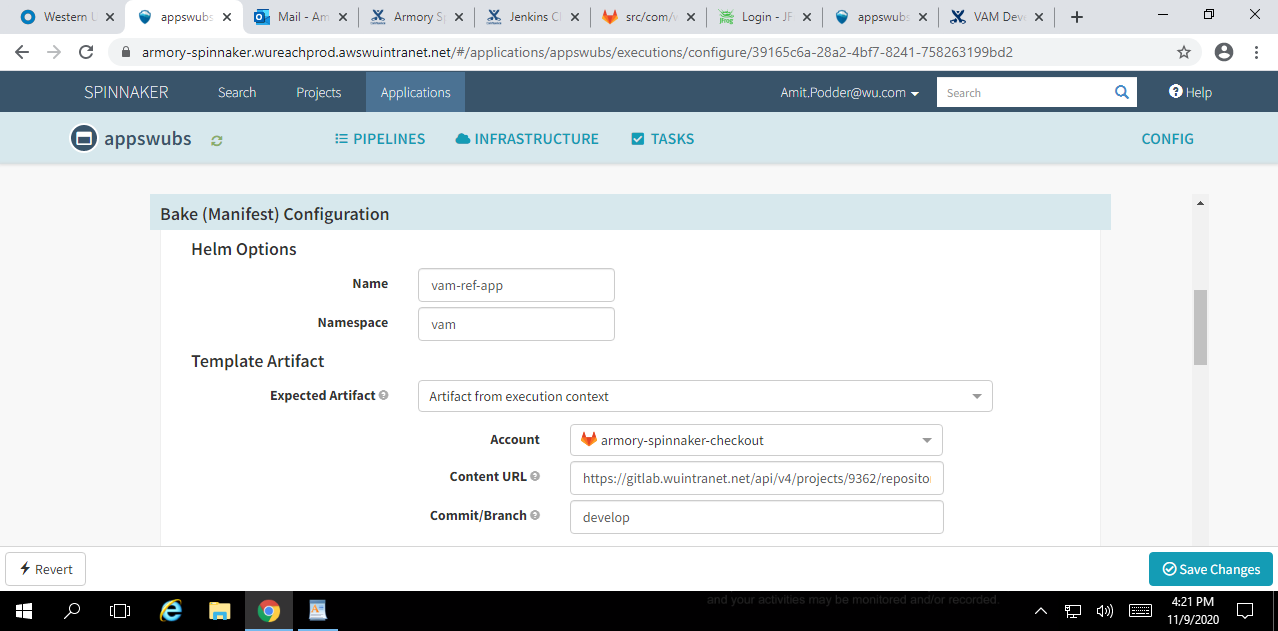
Run As User: Select the user account configured to listen to the Cloudbees Jenkins job. It will be (**sa\_webhook\_pipeline\_executor**)



* **Bake Stage:** Now we add a new stage by clicking on the "Add Stage" option. From the Type dropdown we select "Bake Manifest" and give an appropriate stage name.

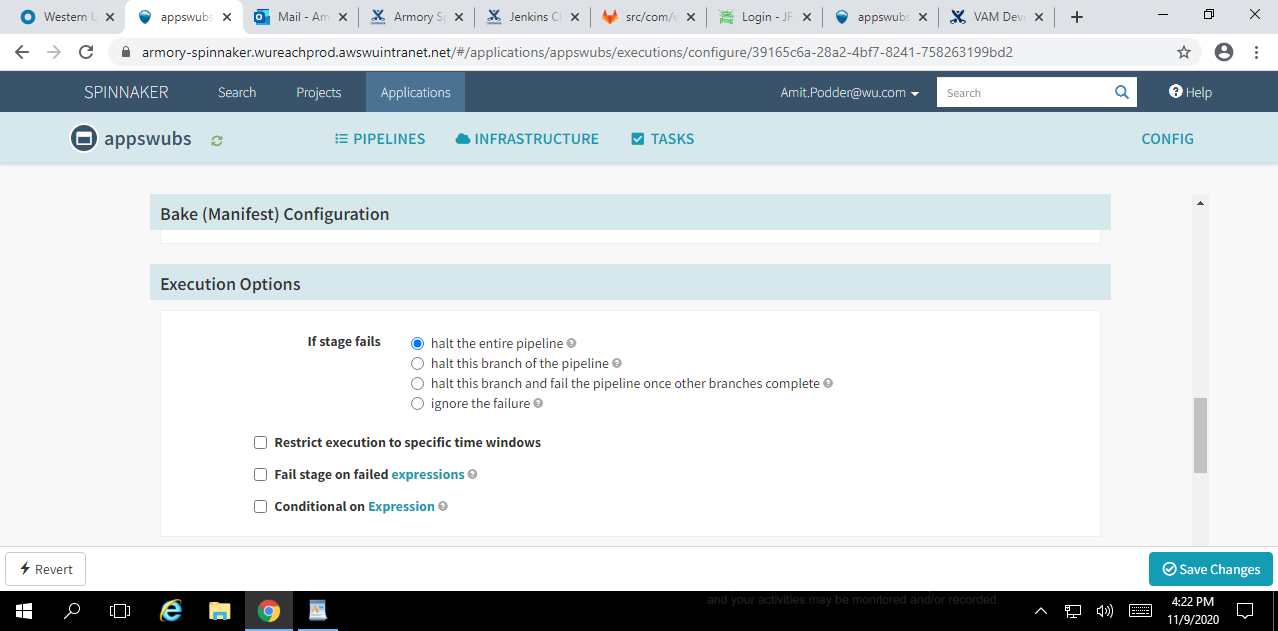


Next we configure the HELM package required to produce the Bake (Manifest) artifacts. We have used HELM3 as render engine for generating Kubernetes specs. Selected the GITLAB repository where HELM package is stored.

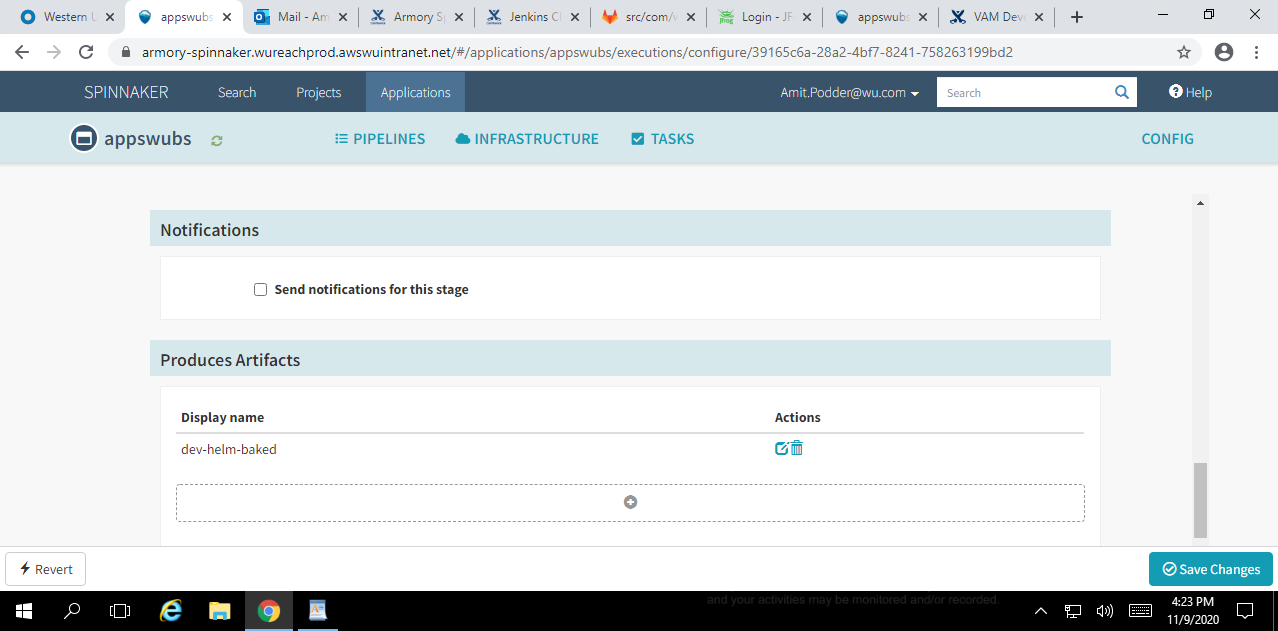


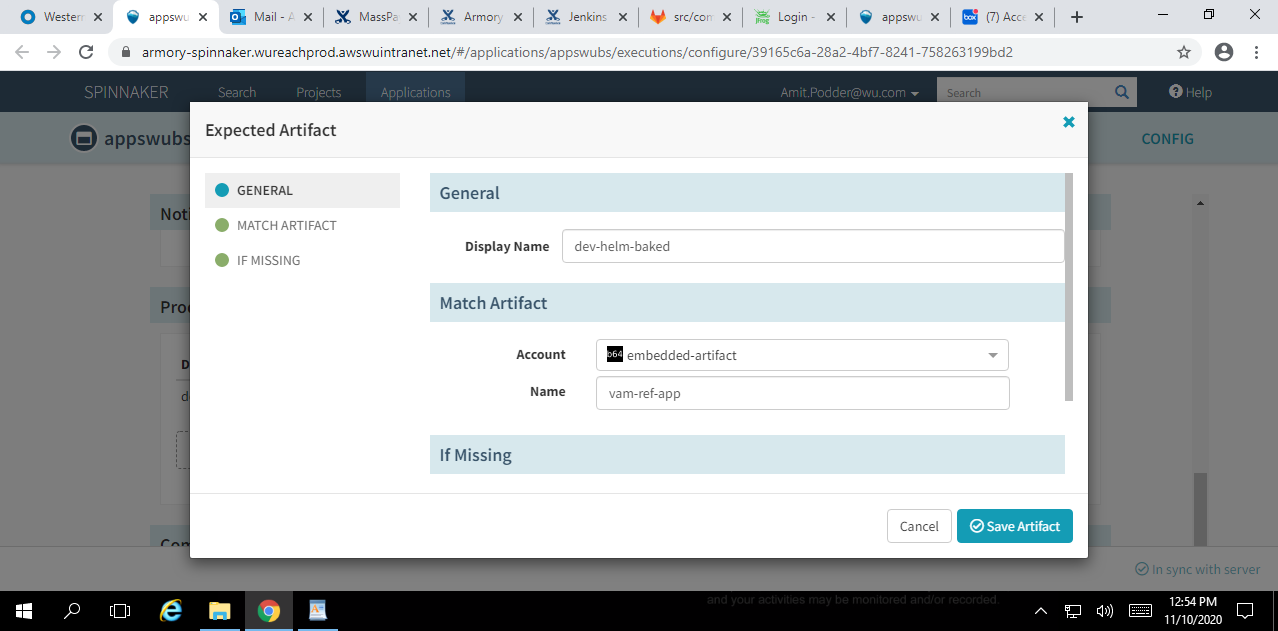
Next we select the option for how the pipeline behaves upon failure of this stage.

We have selected "halt the entire pipeline"



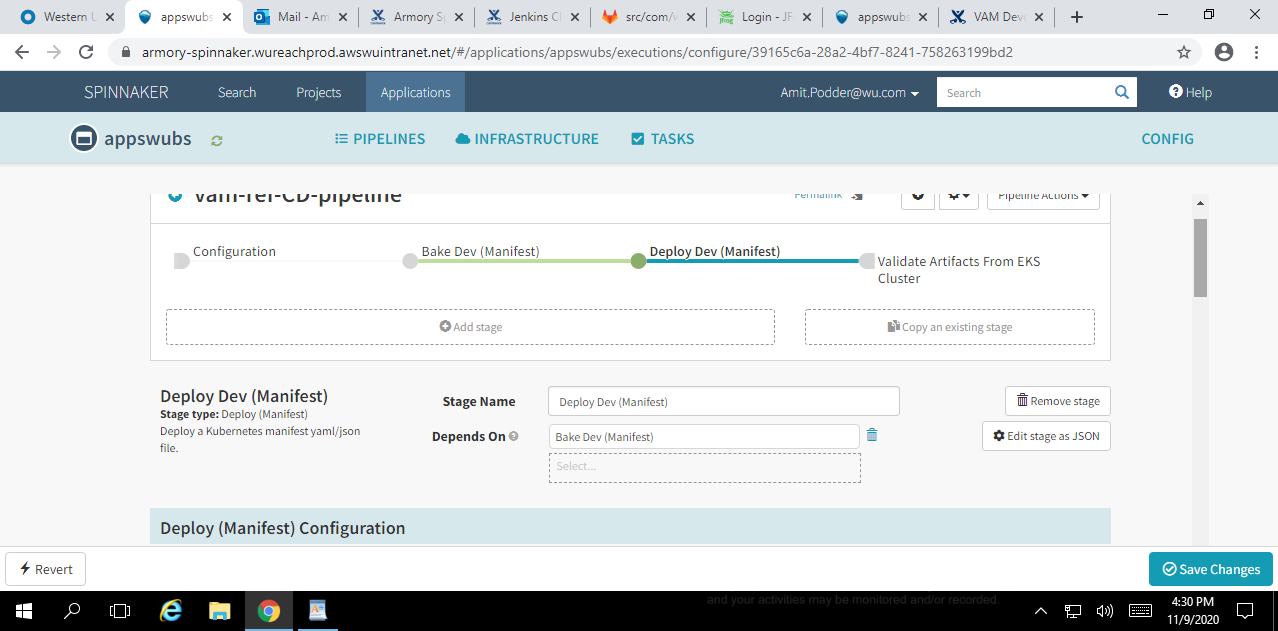
Lastly, we provide the name by which the base64 artifact will produced by the Bake Stage.



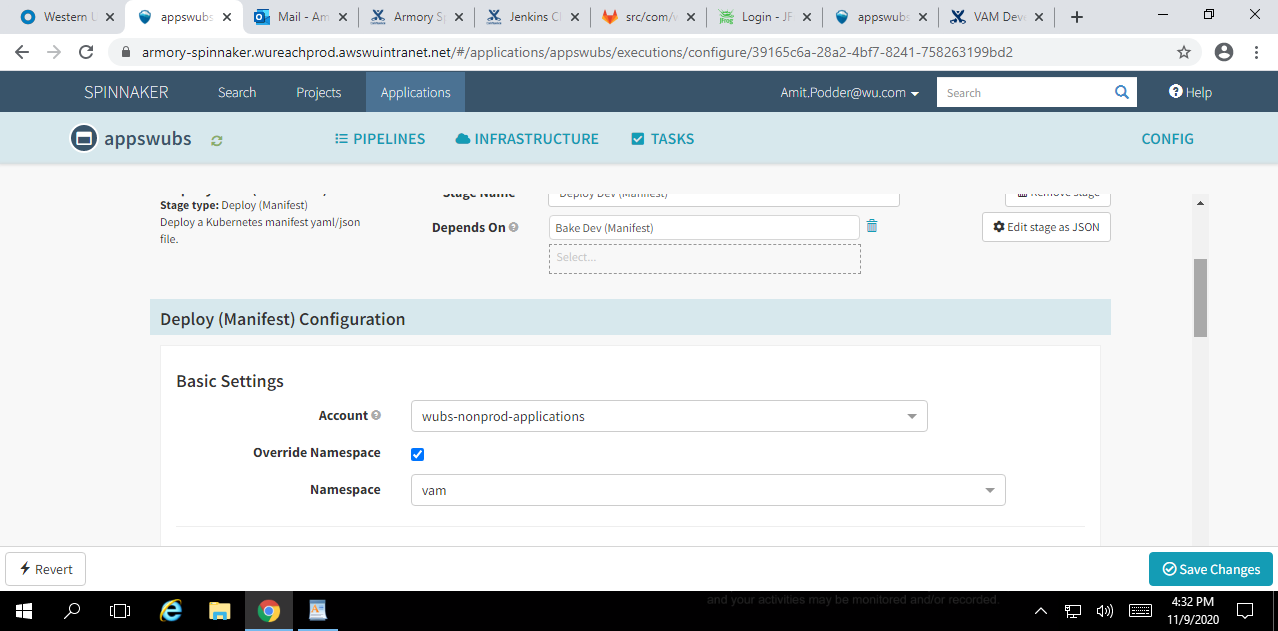


* **Deploy Stage:** This is the stage of our pipeline where we deploy our base64 Artifact prepared from Bake stage onto the EKS cluster.

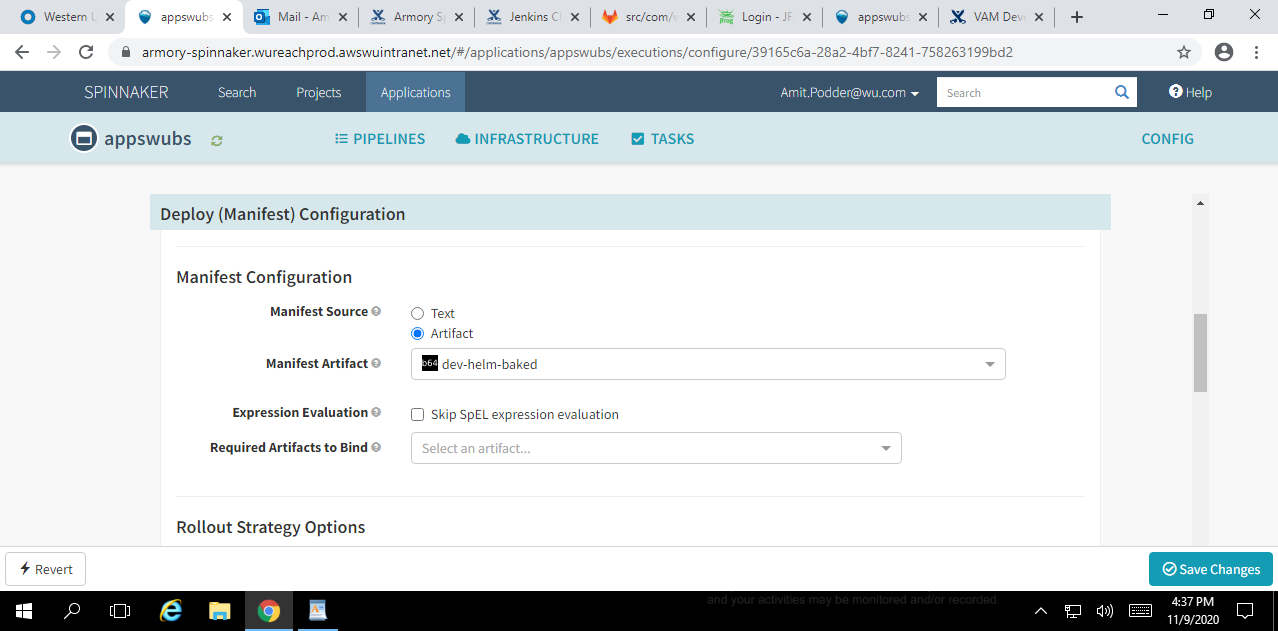
Here we select the "Type" from dropdown as "Deploy" and give a suitable name.



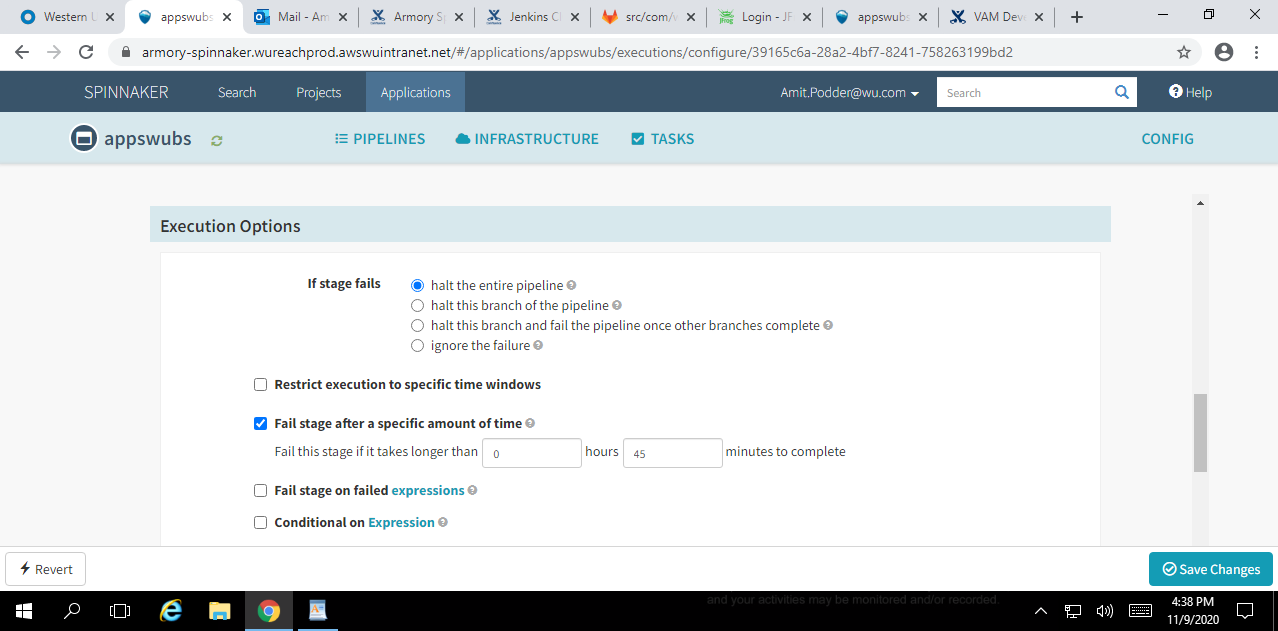
Next we configure the Deploy Manifest. First, we provide the EKS account where we want to deploy our base64 artifact. Below that we provide the Namespace we are using for our pipeline.



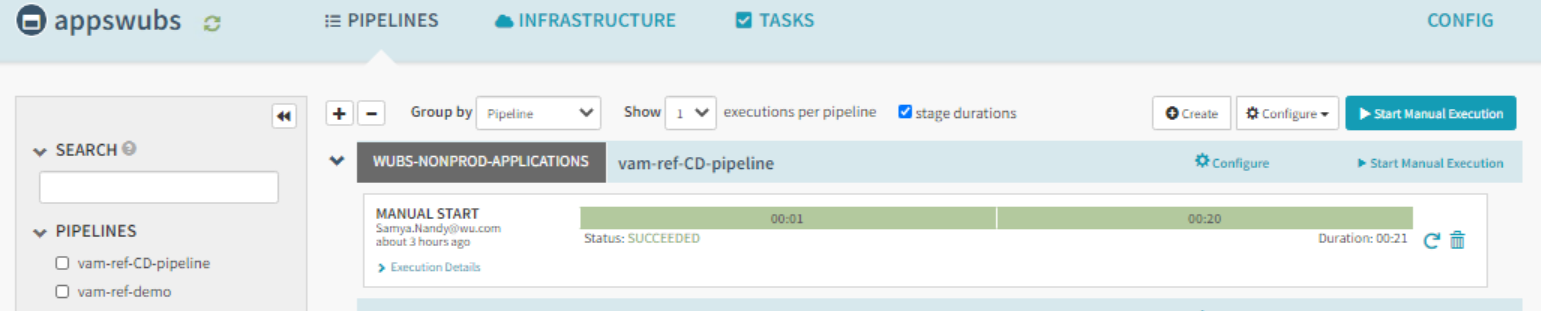
Now comes the configuration where we connect the Deploy stage with the base64 Artifact from the "Bake Stage". Under Manifest Configuration we select "Artifact" as Manifest Source and then select the base64 Artifact in the "Manifest Artifact" dropdown. This will be the same which was produced in the earlier "Bake Stage".



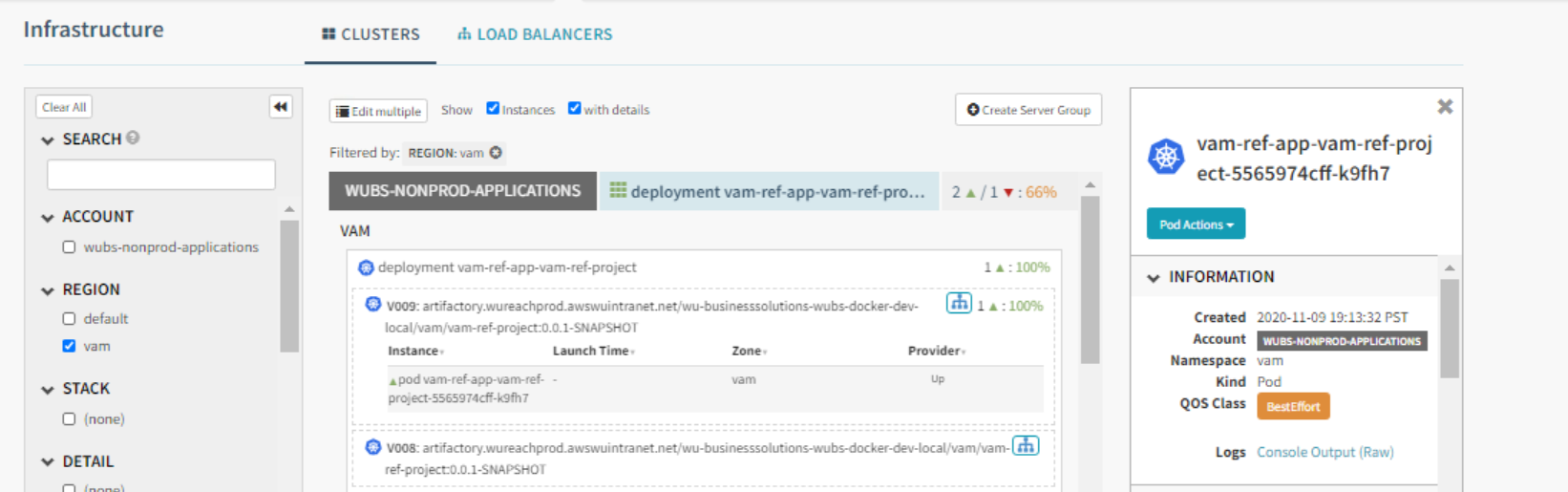
We have provided a Timeout option in this stage. The timeout is set to 45 minutes.



Finally, the pipeline needs to be triggered manually to deploy the artifacts in the EKS cluster. Once the deployment is successful, SUCCEEDED message will be shown.



For successful deployment, we can check the Spinnaker Cluster dashboard. The pods logs can be shown as well from the dashboard.



**Pending Item:**

To access the deployed application from EKS, we need to create the internal load balancer service, Route53 needs to be provisioned for that load balancer service. We need to send the email with all the details to [**TCS-CloudRelTeam@westernunion.com**](mailto:TCS-CloudRelTeam@westernunion.com) to configure the Route53.