

**Intelligent Infra IAAS**

**DevOps Continues Integration – powered by vRealize Code Stream- Build Guide**



Document Control

DOCUMENT REVIEW

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Version | Reviewer | Role | Status |
| 11-Jan-2017 | 1.1 | Shaju Kuttappan | Team Lead | Reviewed |
| 11-Jan-2017 | 1.2 | Shaju Kuttappan | Project Manager | Reviewed |
|  |  |  |  |  |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Change Summary |
| 11-Jan-2017 | 1 | Ram Kumar | Created |
|  |  |  |  |
|  |  |  |  |

Key Contacts for the Private Cloud Factory

| **Role** | **Name** |
| --- | --- |
| Accenture Executive Sponsor | Larry M Socher ([larry.m.socher@accenture.com](mailto:larry.m.socher@accenture.com)) |
| ADC/Private Cloud Offering Lead | Michael Malke ([michael.malke@accenture.com](mailto:michael.malke@accenture.com)) |
| Agile Data Center IDC Factory Lead | Ramesh Nair ([ramesh.nair@accenture.com](mailto:ramesh.nair@accenture.com) ) |
| Cross Solution Product Lead | Jay Ponnuswamy ([jay.ponnuswamy@accenture.com](mailto:jay.ponnuswamy@accenture.com)) |
| Project Manager | Sajith Goopalapuram ([sajith.gopalapuram@accenture.com](mailto:sajith.gopalapuram@accenture.com)) |
| Document Owner | Shaju Kuttappan ([Shaju.kuttappan@accenture,com](mailto:Shaju.kuttappan@accenture,com) ) |

Table of Contents

[1. Executive Summary 4](#_Toc471925177)

[1.1. **Overview** 4](#_Toc471925178)

[1.2. Objective 4](#_Toc471925179)

[1.3. Scope for this Document 4](#_Toc471925180)

[1.4. Intended Audience 5](#_Toc471925181)

[1.5. Assumptions 5](#_Toc471925182)

[2. Product Compatibility and Mapping 5](#_Toc471925183)

[3. Use Case Overview 6](#_Toc471925184)

[3.1. Logical Architecture of the Use Case 6](#_Toc471925185)

[3.2. Use Case Flow 6](#_Toc471925186)

[4. DEPLOY VMWARE CODE STREAM 7](#_Toc471925187)

5. CONFIGURING TOOLS 7

[6. ENDPOINTS FOR A RELEASE PIPELINE](#_Toc471925189) 10

[7. MODELING A RELEASE PIPELINE](#_Toc471925190) 11

8. Use Case Execution 21

[9. Effort Estimate](#_Toc471925191) 22

10. key benefits 22

# Executive Summary

## **Overview**

Accenture uses the Intelligent infra Blueprint to define necessary functional capabilities required to achieve automated solutions.

This blueprint is used to discuss with client what capabilities are necessary, how those capability interrelate, their current maturities, and what trends, technology, and Accenture offering can help evolve to a future state of automated solutions.

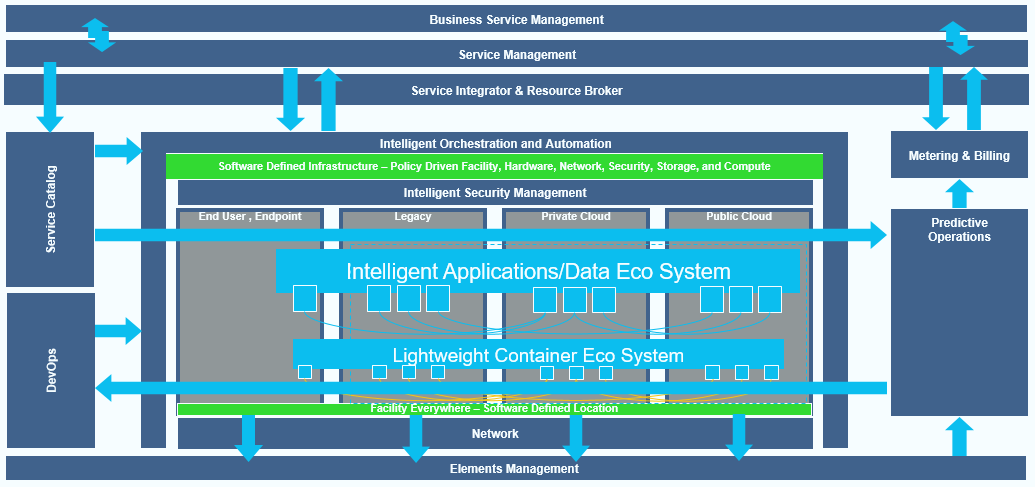


Figure : Inteligent Infra Blueprint

## Objective

The Objective of this document to explain how to build VMware – OpenStack Integration and Cloud provisioning using VMware Integrated Open Stack and vRealize Automation. This use case helps Administrators to reduce time maintaining the access on Open Stack. Users will be able to provision a virtual machine on the fly to Open Stack environment

## Scope for this Document

Scope of this Documents includes

* Use Case Build and Design
* Use Case execution

## Intended Audience

* Cloud Solution Architects
* Data Center administrators
* Infrastructure Transformation team
* Netwrok Architects
* IT consultants

## Assumptions

* VMware vRealize Automation and VMware vRealize Code Stream are installed and configured.

# Product Compatibility and Mapping

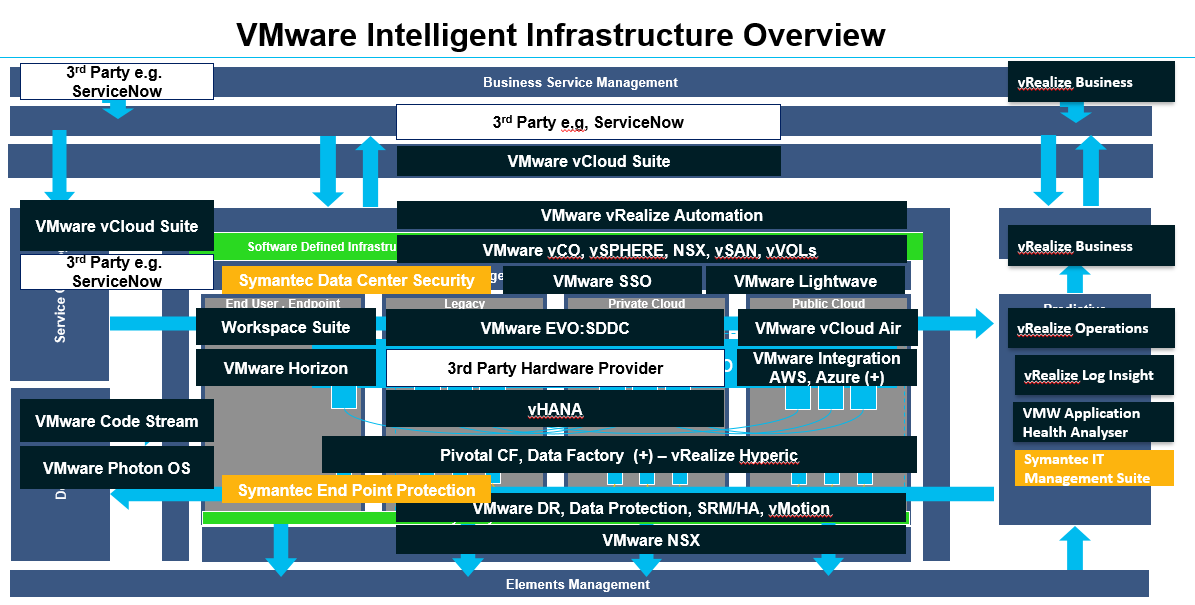
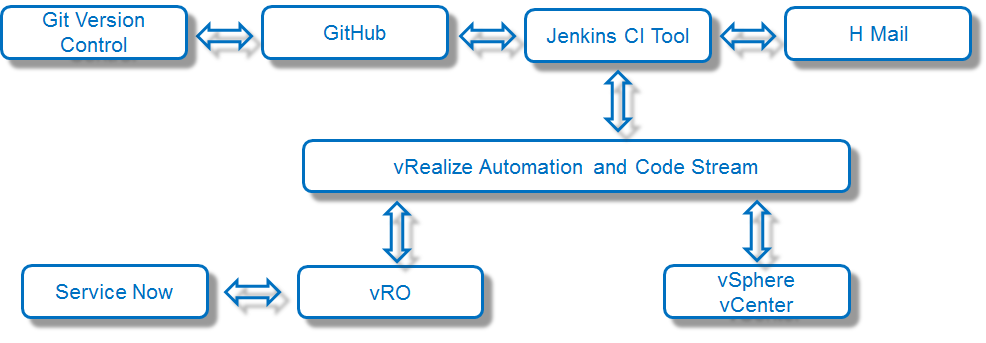


Figure 1: Product Mapping

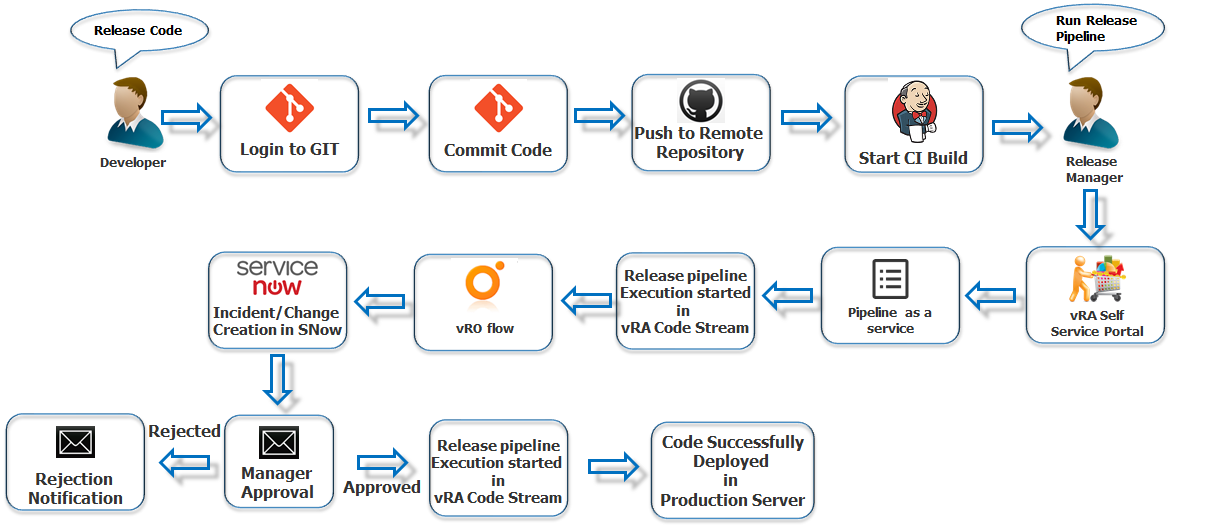
# Use Case Overview

Release Higher Quality Applications Faster While Reducing Operational Risk VMware vRealize Code Stream provides release automation and continuous delivery to enable frequent, reliable software releases while reducing operational risks

## Logical Architecture of the Use Case



## Use Case Flow



# Deploy VMware CODE STream

**Prerequest**

* Deploy and configure a vRealize Automation appliance
* Apply vRealize Code Stream license
* Configure a tenant to assign user roles in vRealize Code Stream

# Configuring Tools

1. **Set up Artifacts Manager**
   1. Download Jfrog Artifacts manager from <https://www.jfrog.com/artifactory/free-trial/#Pro>
   2. Download jfrog-artifactory-pro-5.0.1
   3. Unzip the Artifactory download file to a location on your file system
   4. This will be your *%ARTIFACTORY\_HOME%* location
   5. Define this location as an environment variable called ARTIFACTORY\_HOME.
   6. Requirements
      1. Setting JAVA\_HOME
         1. Make sure that your *JAVA\_HOME* environment variable is correctly set to your JDK installation
         2. This is done by modifying the *JAVA\_OPTIONS* variable in *artifactory.bat*
   7. Manual Installation
      1. Browse to *%ARTIFACTORY\_HOME%\bin* and execute the file *artifactory.bat*
   8. Service Installation
      1. To run Artifactory as a Windows service, browse to *%ARTIFACTORY\_HOME%\bin*, and execute the file *InstallService.bat*
   9. Running Artifactory
      1. After installing Artifactory you need to start the service
      2. To start or stop Artifactory as a service you can use the following command in a Command Prompt window:
         1. Starting and stopping the Artifactory service
            1. ***sc start|stop Artifactory***
         2. Checking the status of the Artifactory service
            1. ***sc query Artifactory***
   10. Getting Started

Artifactory sever URL: <http://localhost:8081>

Follow [Onboarding Wizard](https://www.jfrog.com/confluence/display/RTF/Getting+Started)

* 1. Configuring Repositories

Follow[Artifactoryrepository](https://www.jfrog.com/confluence/display/RTF/Configuring+Repositories), Configure Local Repository

* 1. Local Repository - Name: **CS-DEMO**

1. **Set Up the Repository Server** 
   1. Download the latest [Git for Windows installer](https://git-for-windows.github.io/).
   2. When you've successfully started the installer, you should see the Git Setup wizard screen.
   3. Follow the Next and Finish prompts to complete the installation.
   4. Create New Folder in C:\ “Project”
   5. Right click on Project click on “Git Bash Here” (It will open with Bash Command line window)
   6. Check Git version with below command

git –version

* 1. Run the following commands to configure your Git username and email using the following commands

$ git config --global user.name "USERNAME"

$ git config --global user.email”USERNAME@DOMAIN.COM”

* 1. Executing git init creates a .git subdirectory in the project root, which contains all of the necessary metadata for the repo.

$ git init

* 1. Login to <https://github.com/> signup with new account, for remote repository in public.
  2. Once after log in to GitHub portal create a New Repository.
     1. Repository name
     2. Select Public
     3. Check Initialize this repository with a README
  3. Now we have to clone this remote repository to local repository
     1. Copy Clone with HTTPS <https://github.com/ramrk22/CodeStream.git>
     2. Login to Git Server go to C:\Project right click Git Bash Here
     3. Execute Git command to clone

$ git clone [https://github.com/ramrk22/CodeStream.git](https://github.com/ramrk22/TestGet.git)

* + 1. Now you can see files added in to folder in Project
    2. Now we can create a sample project for best exercise
       1. Go to Project folder create index.html with sample program codes
       2. Open GIT Bash Here
          1. $ git status
          2. $ git add -A
          3. $ git commit -m “Commit Message”
          4. $ git push -u origin master
          5. Now you can see index.html file in remote repository <https://github.com/ramrk22/CodeStream.git>.
          6. To revert commit we should check commit logs using$ git log
          7. You see like this
          8. Select commit which we need to revert and copy the commit unique number
          9. $ git reset –Hard 48963d
          10. $ git push -f origin master

1. **Set up Jenkins Server** 
   1. Download Jenkins for Linux machine <https://jenkins.io/>
   2. Installing Jenkins as a [Windows Service](https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+as+a+Windows+service)
   3. Now login to <http://localhost:8080>
   4. Jenkins will startup processed with Wizard
      1. Admin login with default password
      2. Plugin installation – select Suggested Plugin’s proceed installation complete
      3. Create admin user
      4. Click on Manage Jenkins -> Configure System-> Artifactory-> Artifactory servers: Enter Artifactory server details in required filed
      5. Click on Manage Jenkins -> Global Tool Configuration -> Git -> Git Installations -> Path to Git executable “C:\Program Files\Git\bin\git.exe”
      6. Apply and Save
      7. Now click on Create New Jobs
         1. General -> Name and Details
         2. Source Code Management -> Select **Git**
            1. Repositories URL: <https://github.com/ramrk22/CodeStream.git>
            2. Additional Behaviors: Local subdirectory for repo -> **CS-Demo**
            3. Build Triggers -> Check Box on **Poll SCM** -> Schedule: \* \* \* \* \* (it represent that build trigger every "every minute")
            4. Build Environment -> Check Box **Generic-Artifactory Integration**
            5. Artifactory Configuration

Download and upload by -> Check Legacy patterns (deprecated)

Artifactory upload server -> <http://10.211.203.228:8081/artifactory>

Target Repository -> **CS-Demo**

Artifactory Credential

Published Artifacts -> ***CS-Demo-$BUILD\_NUMBER.tar.gz***

* + - * 1. Build

Execute Shell

*#!/bin/bash*

*rm -rf CS-Demo.tar.gz*

*tar -zcvf CS-Demo.tar.gz CS-Demo*

*tar -zcvf CS-Demo-$BUILD\_NUMBER.tar.gz CS-Demo*

* + - * 1. Post-build Actions

Archive the Artifacts

File to Archive - > ***CS-Demo.tar.gz,CS-Demo-$BUILD\_NUMBER.tar.gz***

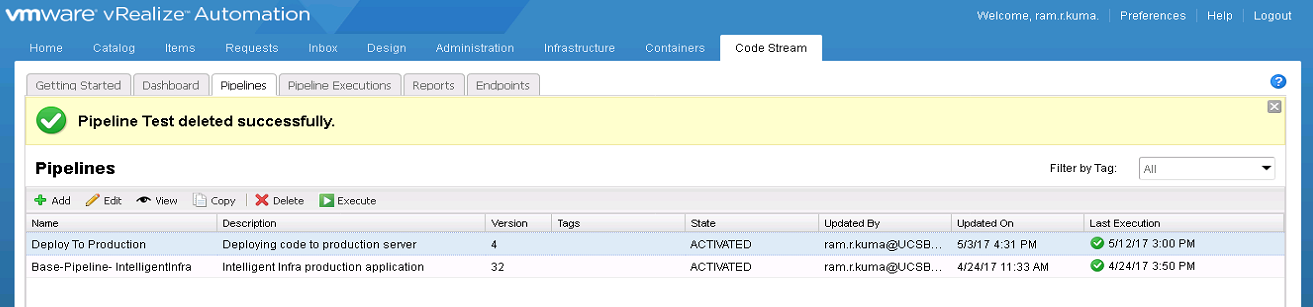
* + - * 1. Apply and Save

# Endpoints for a Release Pipeline

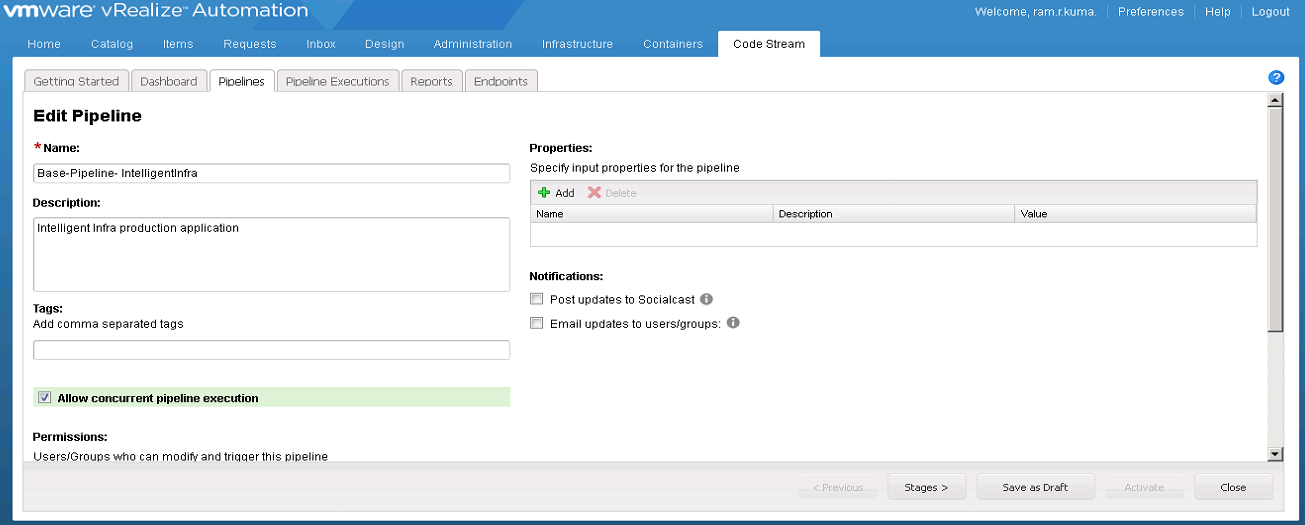
1. Register an Artifactory Server Endpoint
   1. Click **Add**, and from the Endpoint Type drop-down menu select Artifactory.
   2. Enter an **Artifactory server endpoint name** and an applicable description.
   3. Enter the Artifactory server configuration details, and click **Save**.
2. Register a Jenkins Server Endpoint
   1. Select **Code Stream > Endpoints.**
   2. Click **Add**.
   3. Select Jenkins from the Endpoint Type drop-down menu.
   4. Enter a Jenkins server endpoint name and an applicable description.
   5. Enter the Jenkins server configuration details.
      1. **User Credentials**
      2. **Folder Path**
      3. **URL**
      4. **Polling Interval**
      5. **Request Retry Count**
      6. **Retry Wait Time**
   6. When the endpoint is created, you can select **Offline Creation** to require a validation and certificate acceptance.
   7. Click **Save**.
3. Register a vRealize Automation Server Endpoint
   1. Select **Code Stream > Endpoints.**
   2. Click **Add**.
   3. Select the **vRealize Automation** version from the Endpoint Type drop-down menu.
   4. Enter a vRealize Automation endpoint name and an applicable description.
   5. When the endpoint is created, you can select **Offline Creation** to require a validation and certificate acceptance.
   6. Click **Save**.
4. Register a vRealize Orchestrator Server Endpoint
   1. Select **Code Stream > Endpoints.**
   2. Click **Add**.
   3. Select **vRealize Orchestrator External** from the Endpoint Type drop-down menu.
   4. Enter a vRealize Orchestrator server endpoint name and an applicable description.
   5. Enter the vRealize Orchestrator server configuration details.
      1. **User Credentials**
      2. **URL**
      3. **Tenant**
   6. Click **Save**.

# Modeling a Release Pipeline

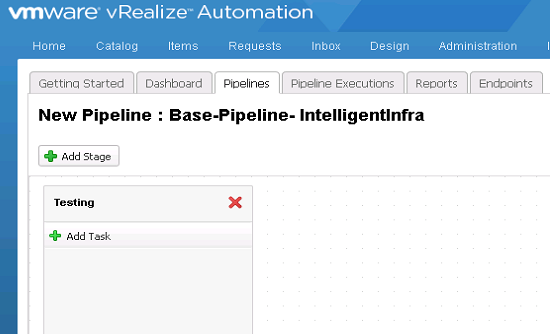
1. Click the **Code Stream** tab.



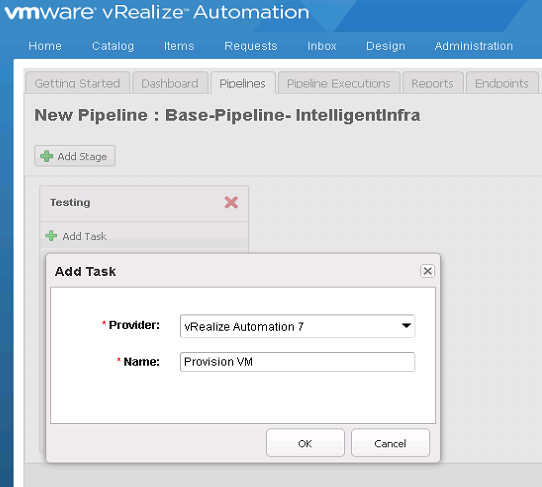
1. On the Pipelines tab, select **Base-Pipeline- IntelligentInfra** pipeline and click **Edit** to configure it.



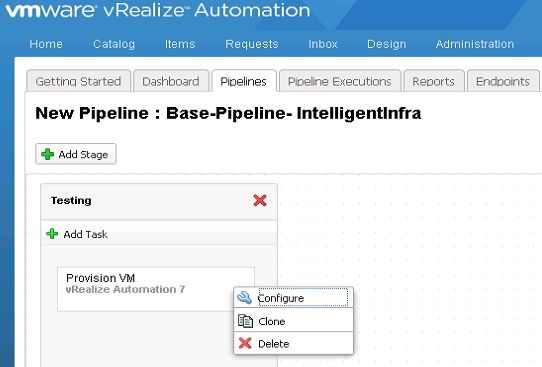
1. Now click on **Add Stage** and name it **TESTING**



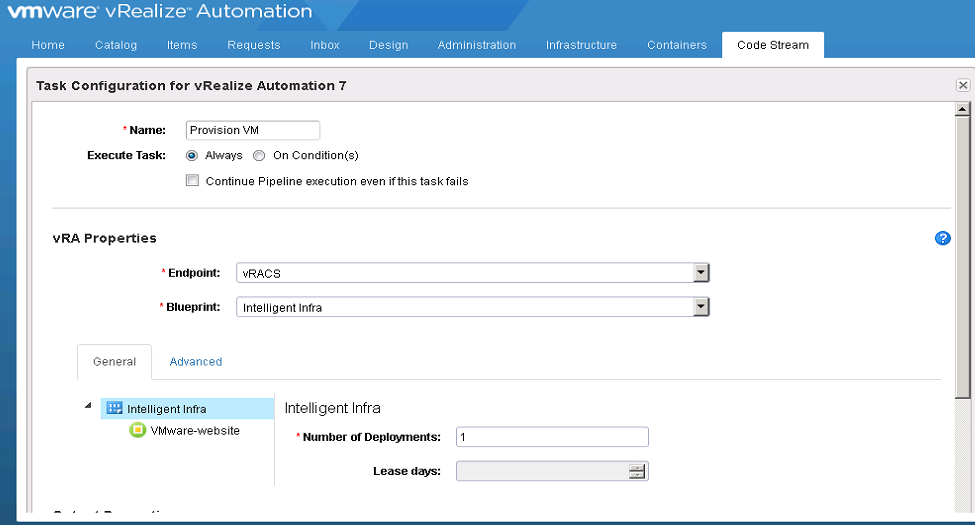
1. Click **Add Task**, select **vRealize Automation 7** from the Provider drop-down menu.



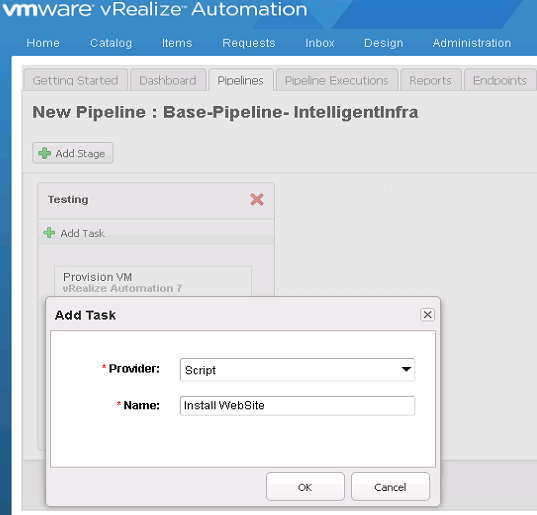
1. Click on Settings and Configure.



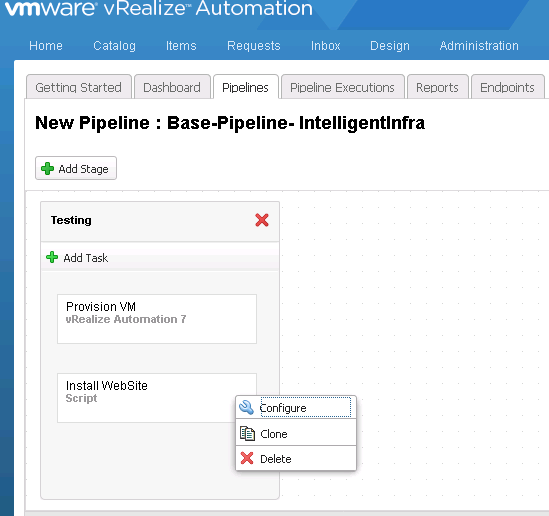
1. Select the endpoint **vRACS** and the blueprint **IntelligentInfra** to task configuration for vRA Automation 7.



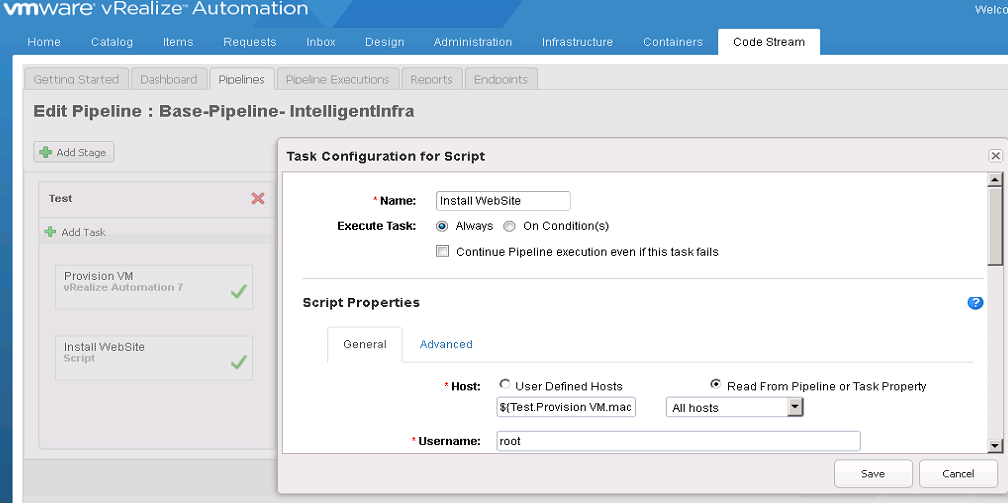
1. Click **Add Task**, Select **Script** from the Provider drop-down menu.



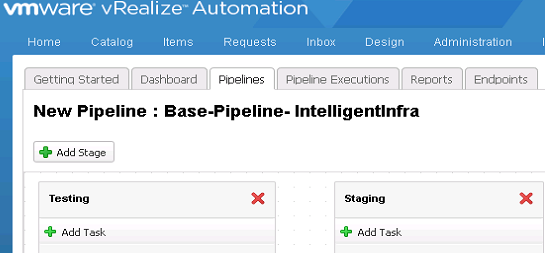
1. Click on Settings and Configure.



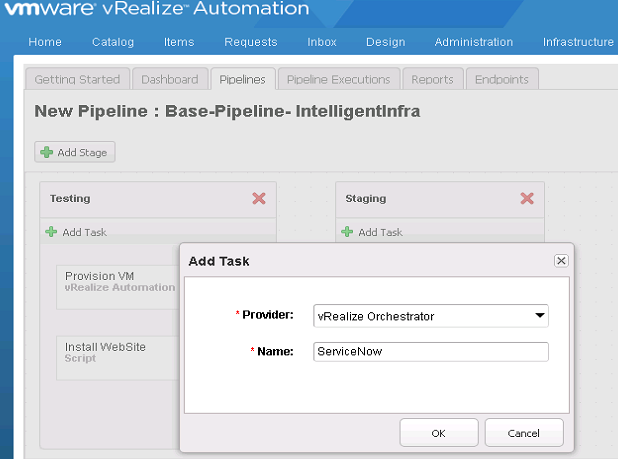
1. Task Configuration for **Script**



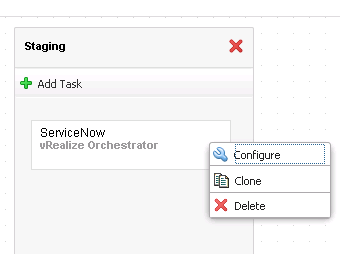
1. Now click on **Add Stage** and name it **STAGING**



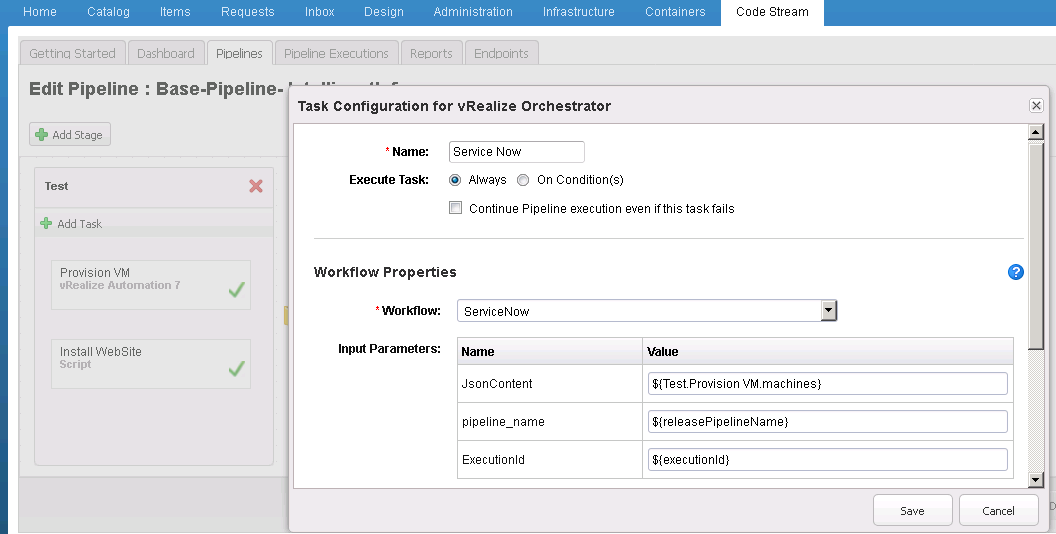
1. Click **Add Task**, Select **vRealize Orchestrator** from the Provider drop-down menu.



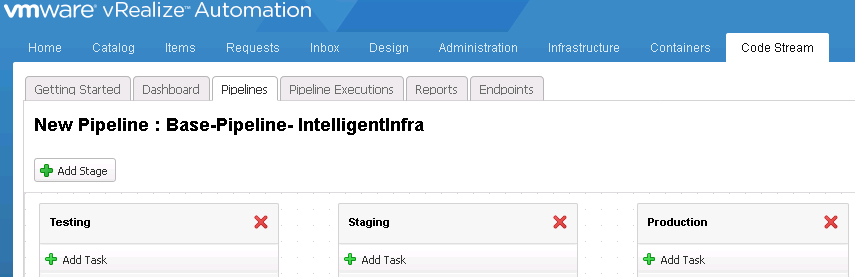
1. Click on Settings and Configure



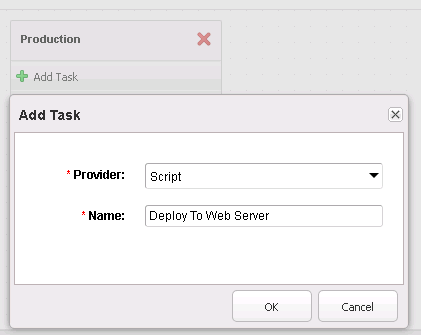
1. Task Configuration for **vRealize Orchestrator**, Select Workflow **Service Now** and enter all Input Parameters.



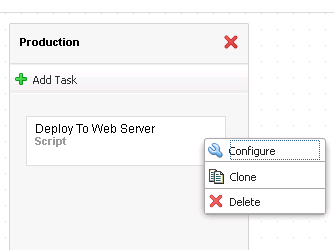
1. Now click on **Add Stage** and name it **PRODUCTION**



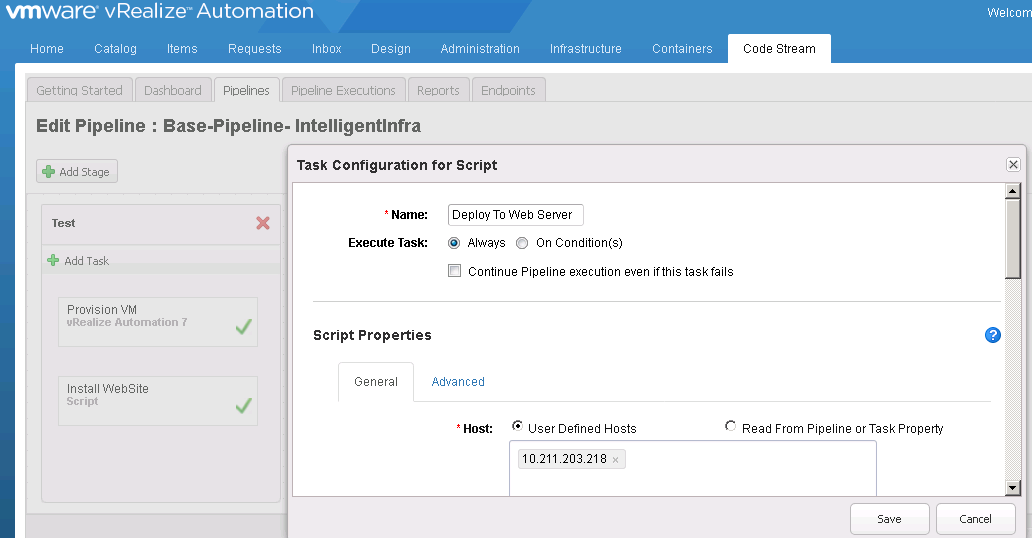
1. Click **Add Task**, Select **Script** from the Provider drop-down menu.



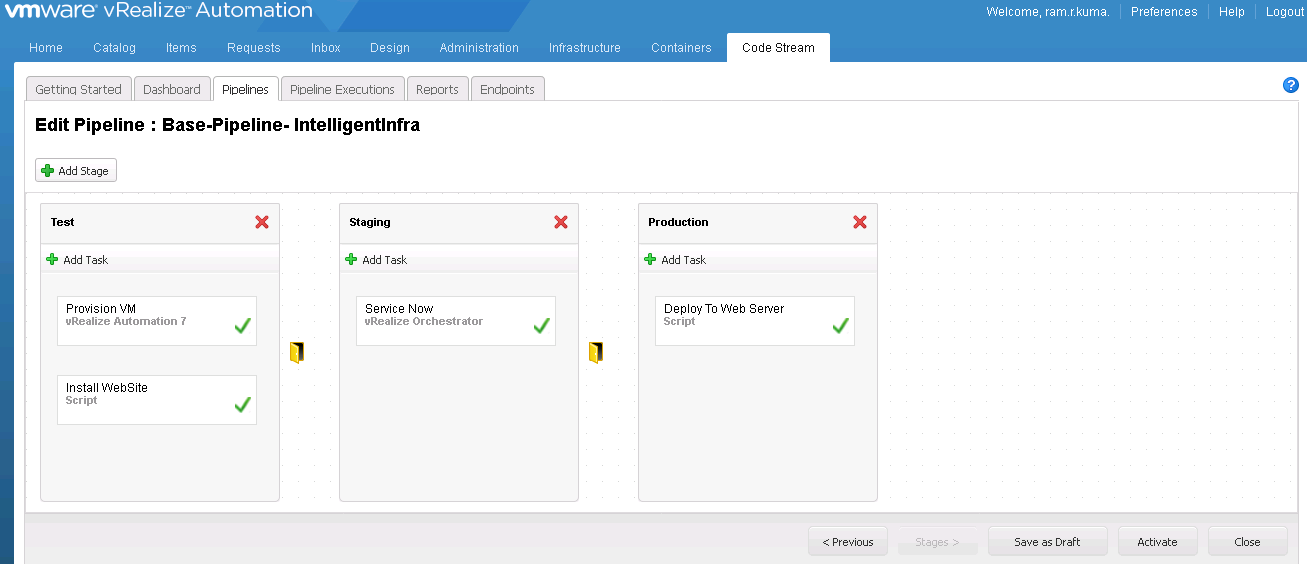
1. Click on Settings and Configure



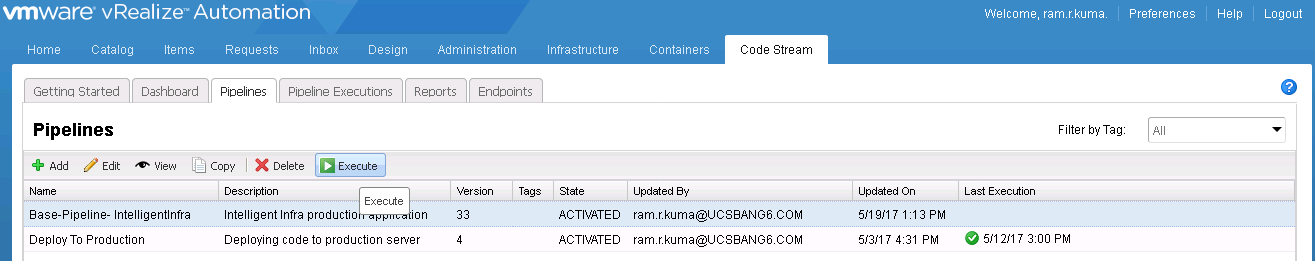
1. Task Configuration for **Script**, define **HOST NAME** of production application server and other configuration details.



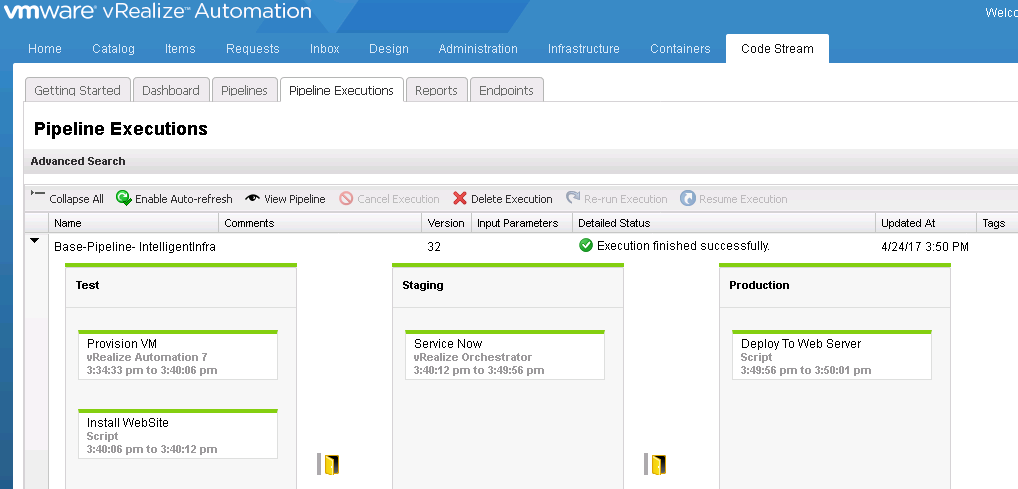
1. Once all Stages and Task are configured **SAVE** and **ACTIVETE** release pipeline.



1. Click > **Activate** the pipeline, and click **Execute**



1. After Release Pipeline executed successfully, pipeline execution status shown as below.



# Use Case Execution

1. Login to Devloper System.
2. Go to > C:\CodeStream Folder.
3. Right click “index.html” file open with text editor.
4. Update Login Form code at line 60 and save and close.
5. Right click inside C:\CodeStream folder click Git Bash Here.
6. Commit Code in Git and push to Git Hub Repository.
7. Login in to vRA <https://vRA>
8. Click on catalog tab.
9. Click on “Release Pipeline” Catalog Item.
10. Provide description and click Submit.
11. Click Code Stream tab.
12. Click Pipeline Execution, where you see pipeline execution task stage by stage.

# Effort Estimate

|  |  |  |
| --- | --- | --- |
| **DevOps Continues Integration Use Case Development** | | |
| Effort Estimation | | |
| Activities | | Man Hours |
| 1 | Planning | 12 |
| 2 | vRealize Automation 7.2 installation and configration | 10 |
| 3 | Tools Configrations | 2 |
| 4 | Configruing Git Version | 3 |
| 5 | Configruing GitHub | 1 |
| 6 | Configruing Jenkins | 10 |
| 7 | Configruing Artifacts Manager | 9 |
| 8 | Configruing vRO | 2 |
| 9 | vRA Code Stream Configrations | 5 |
| 10 | Endpoint Configrations | 2 |
| 11 | Release Pipeline Configration | 4 |
| 12 | Testing | 3 |
| 13 | Documentation | 48 |
| 14 | Execution | 3 |
| Total | | 114 |

# **Key Benefits**

* Continuous Delivery or DevOps initiatives become more productive
* Deliver applications faster
* Deliver more reliable applications
* Leverage existing tools and processes
* Improve governance and visibility
* More Manual Efforts, Large Team Co-Ordination and Time Consuming Process.