Version 0.2

21st NOV 2017

Capgemini Immediate

Deploying HID-Release-B application in AWS Environment – Packer & Terraform

Table of Contents

[1 Introduction 3](#_Toc485582919)

[2 Pre-Requisite 4](#_Toc485582920)

[3 Setting up the environment in AWS 5](#_Toc485582921)

4 Packer Activity - Image Builder (create pipeline) 7

[5 Packer Activity - Image Builder](#_Toc485582923)  9

6 Packer Activity - Image Builder (Run pipeline) 11

[7 Terraform Activity – Provisioner (Create pipeline)](#_Toc485582925)  12

[8 Terraform Activity – Provisioner (Run pipeline)](#_Toc485582926) 14

[9 Document Version 1](#_Toc485582927)5

# Introduction

This document outlines the steps involved in setting up below HID-WAS application in AWS environment server using packer image & Terraform scripts.

* HID- Release B

# Pre-Requisite

* GoCD Agent which has connectivity with the AWS-POC environment
* Access to AWS Jumpserver – 52.214.120.0
* Access to AWS console
* Access to bootstrap server – 172.31.30.250
* Acccess to GoCD, Nexus and Gogs in the AWS Environment

URL::

Gogs 🡪 <http://172.31.30.250:3000/>

Nexus🡪 <http://172.31.30.250:8081/nexus>

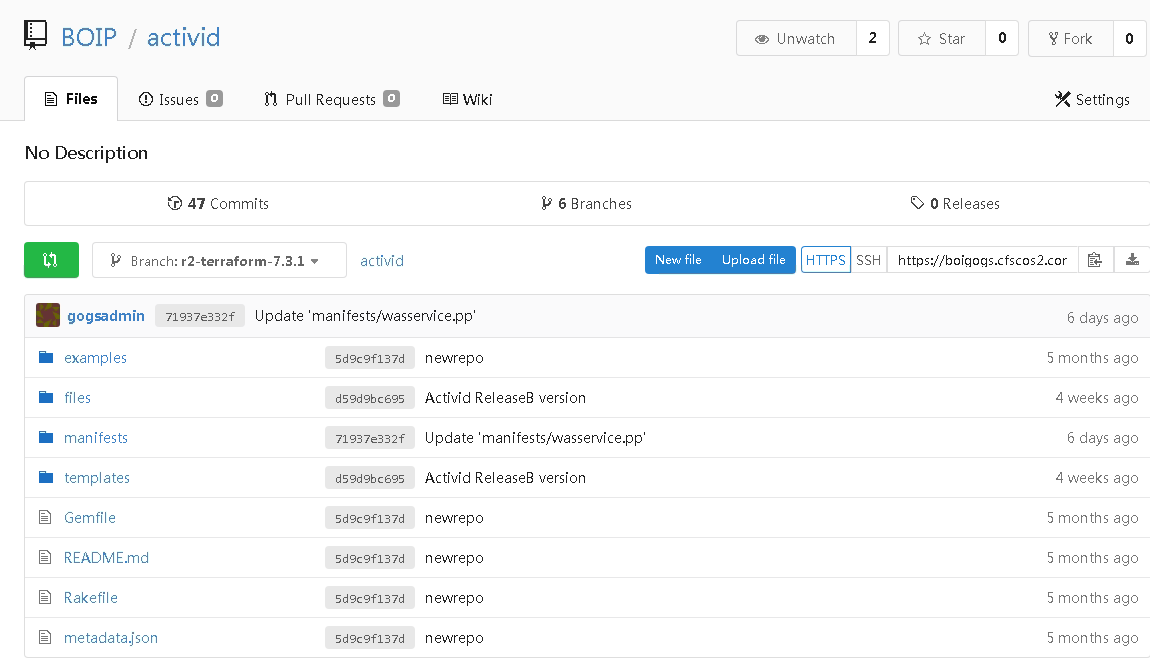
GoCD🡪 <http://172.31.30.250:8153/go/>

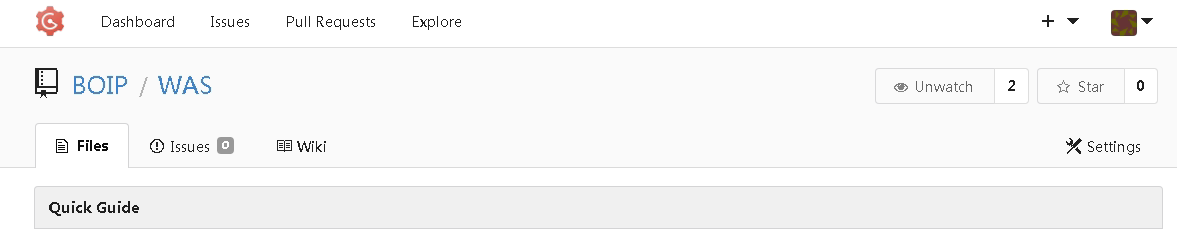
Consul🡪 <http://172.31.30.250:8500/ui/>

# Setting up the environment in AWS

We need to move all the binaries & modules related to ActivID from Bitbucket/Nexus to AWS Gogs & AWS Nexus.

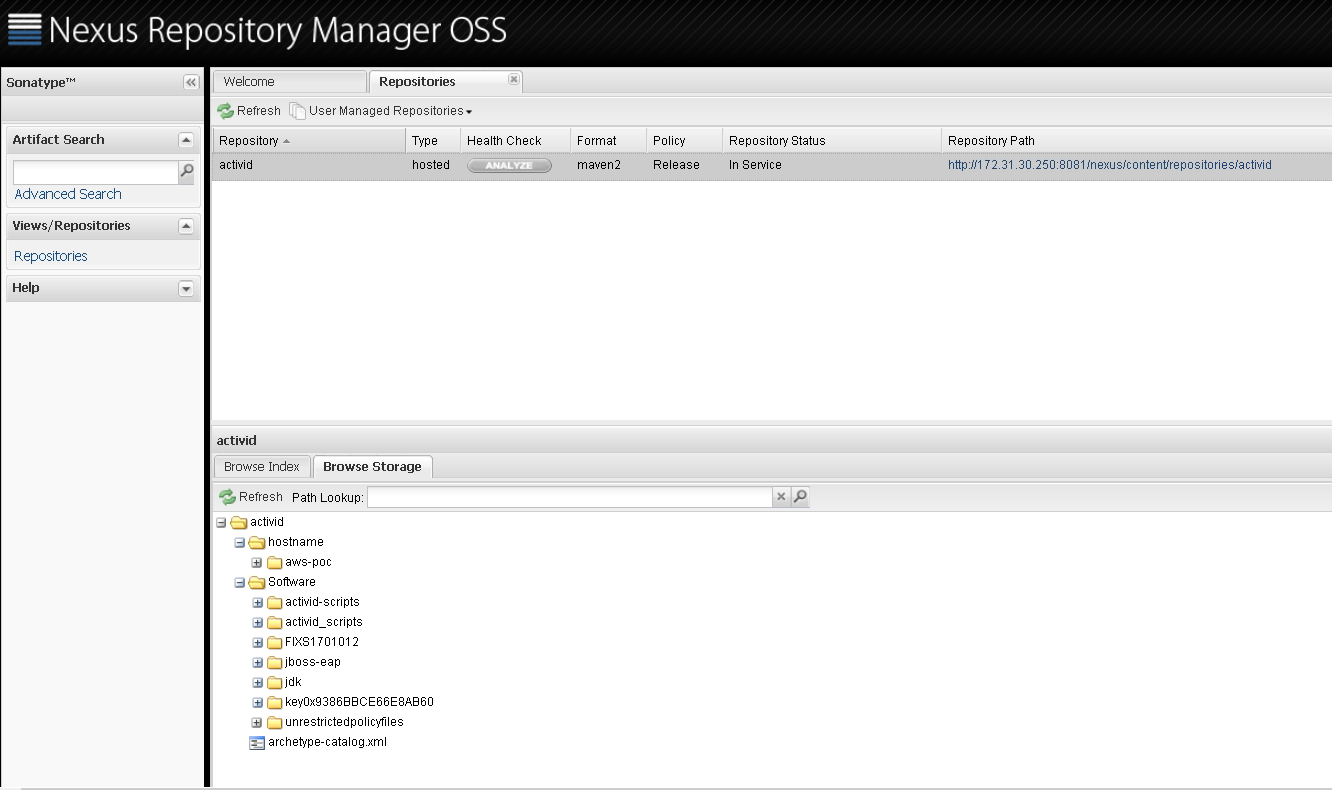
* Clone the ActivID & WAS module from the bitbucket and move the folders to AWS Jumphost.

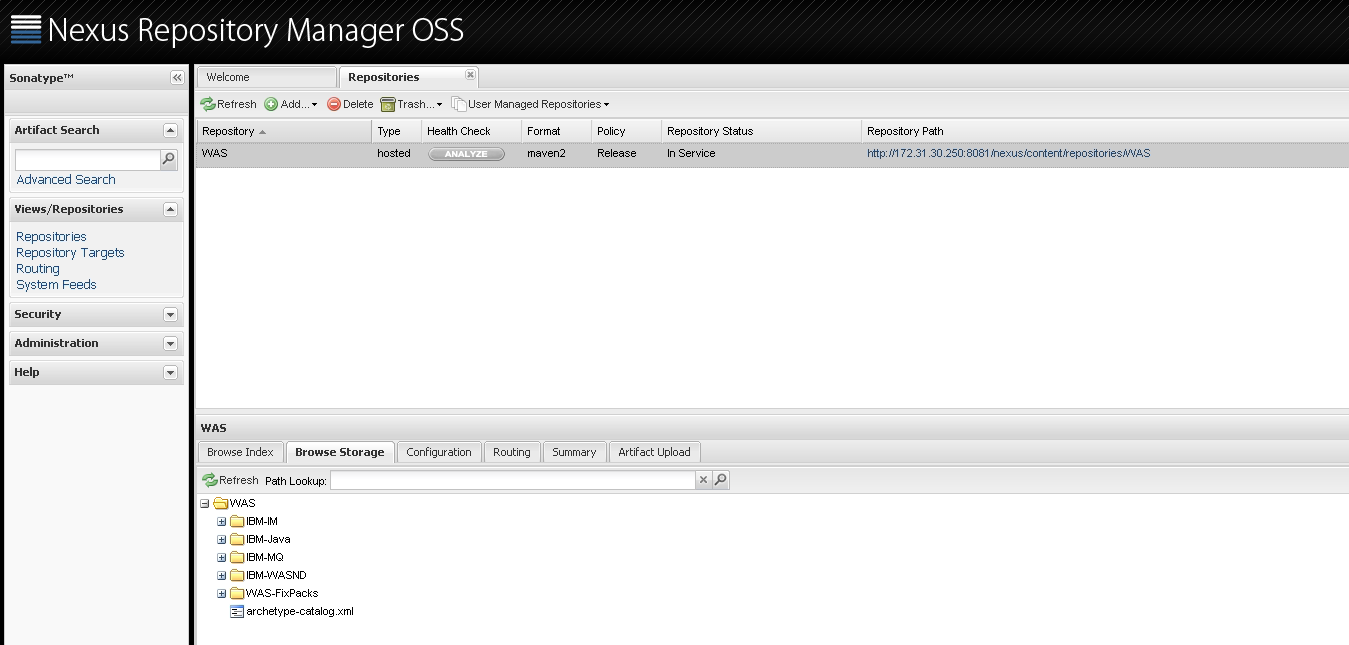




* Create the empty repositories for ActivID & WAS in AWS Gogs (172.31.30.250:3000) and commit the copied folders to the respective repositories.
* Download the binaries for ActivID & WAS from infra nexus and upload to AWS S3 bucket through UI console. Now login to the AWS Jumpserver & login to s3 console and download the binaries locally.
* Now upload the binaries to AWS Nexus(172.31.30.250:8081) with proper folder structure similar to infra nexus.

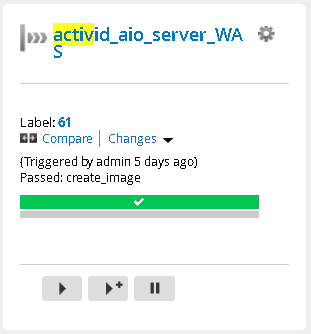
Now we have the similar setup as of infra tooling.



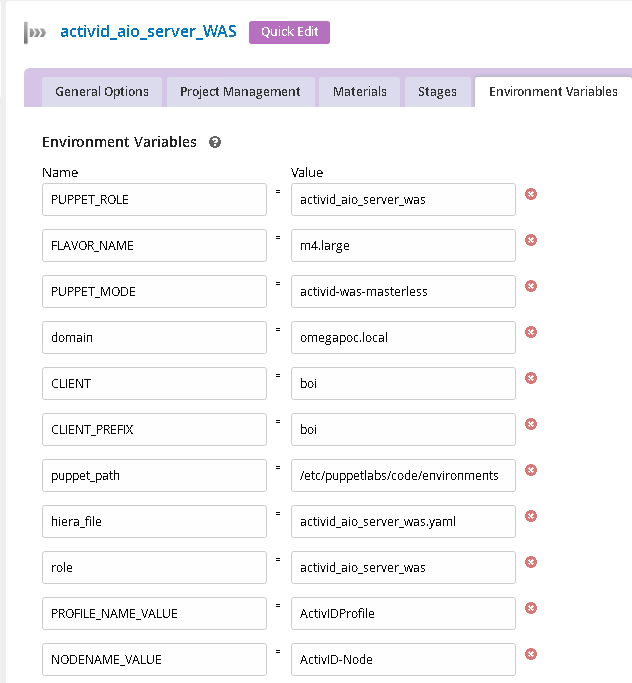
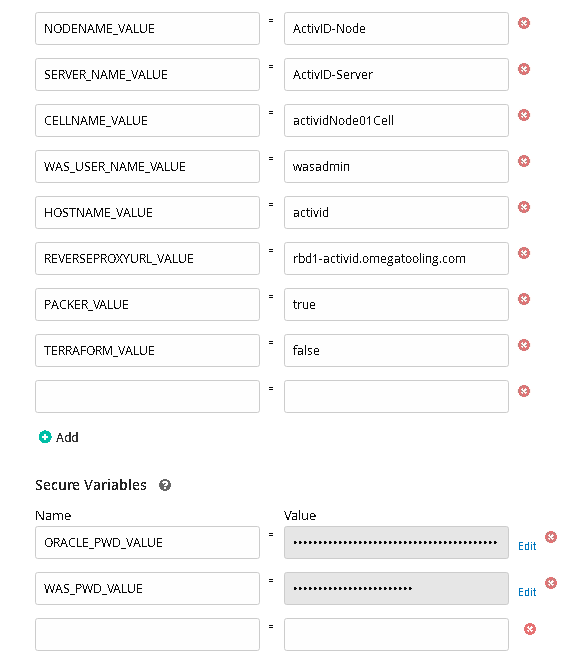


# 4 Packer Activity – Image Builder (Create Pipeline)

* Clone the existing pipeline of HID-RELEASE A and name it as “ActivID-aio-server-was”



* The cloned pipeline will have the exports w.r to AWS such as (access key, secretkey, flavour etc). So we need to import the app related env variables.
* Import all the environmental variables w.r to HID-RELB , below are the variables which we have imported w.r to RelB

* Create a new template for RelB and add stages & jobs similar to RelA template , the order of the task execution should be similar to the RelA template .
* Alter the Materials & Artifacts of the ActivID RelB application pointing to 172.31.30.250 – AWS environment.
* Create a JSON file activid-aio-server-was-masterless.json which will be passed as an input to the packer pipeline. (take RelA HID/temenos JSON file as reference).

The JSON file will contain the environment variables & user variables declared and also the sell commands for copying the puppet modules, and executing the puppet run etc.

Below is a sample JSON file for reference.



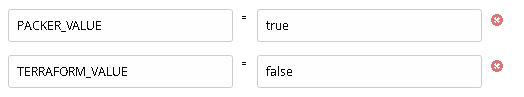
# 5 Packer Activity – Image Builder

* Now as per the requirement we have to make the changes in the puppet code as well.
* The code customization includes which part of the puppet code should be executed as a part of packer image build & which part should be included in the terraform run (user-data).

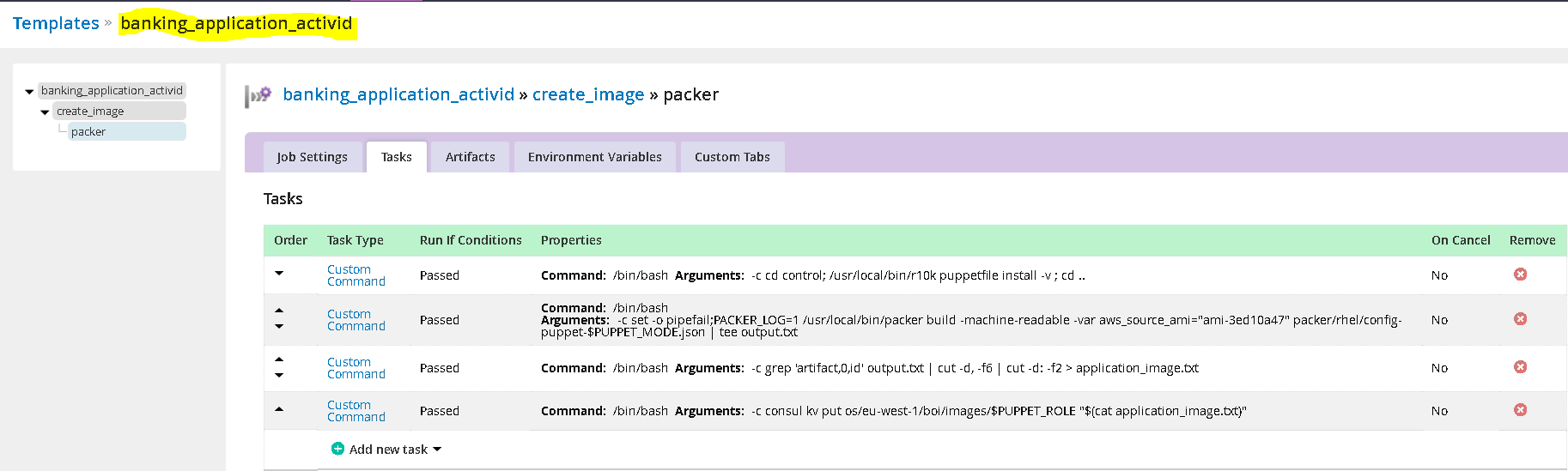
In RELB ActivID, we have 5 classes in puppet.

* 1. Was.pp
  2. Installables.pp
  3. Configure.pp
  4. Wasconfigure.pp
  5. Wasservice.pp
* In packer image build, we have included the was.pp, installables.pp & configure.pp and some exec and file resources of wasconfigure & wasservice.pp
* The skipped parts of these modules in puppet execution will be carried out as a part of user data in Terraform activity.

Now the puppet code is modified with a 2 new variables $packer & $terraform.



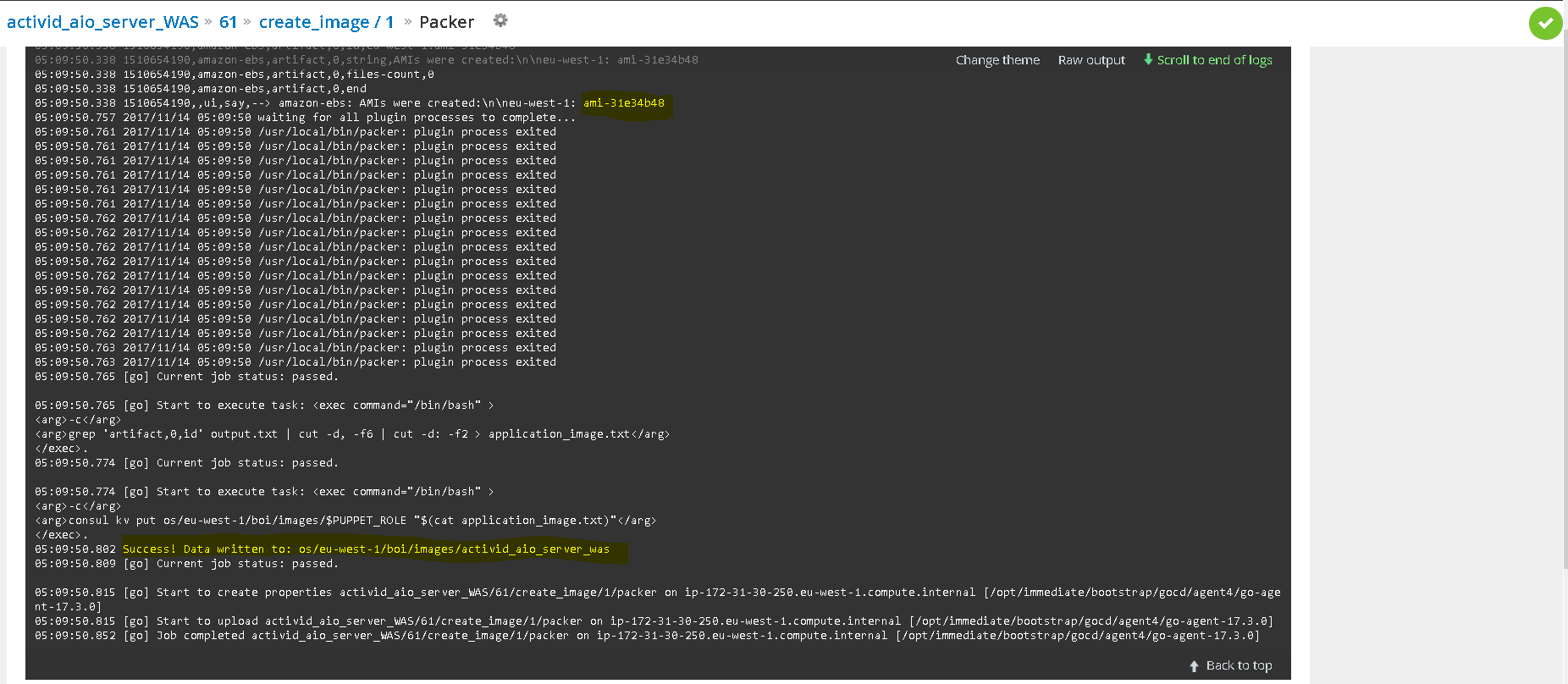
* + If $packer=true, then it will execute the parts which don’t have run-time / environment dependencies.
  + If $terraform=true, then it will execute the remaining part which was excluded in packer activity.
  + If we give both $packer & $terraform variable as true, then it will run the entire puppet code in a single run.
* Now, we have the pipeline created & the environmental variables declared in the pipeline. Next, we have to create the template & add the stages,jobs & tasks and attach the template to the pipeline.
* Here, we have created a template “banking-apllication-activid” & added the tasks for this as below.



# 6 Packer Activity – Image Builder (Run Pipeline)

* Once the pipeline has been created & configured and the customized code has been committed into Gogs, then we can proceed to trigger the pipeline run.
* Upon successful run, this will create a packer image (ami-xxxxxxx) in the AWS environment.
* This is the image we have to use in the terraform activity and pass the run time variables in terraform pipeline.

Below is the output of a successful packer-builder run.

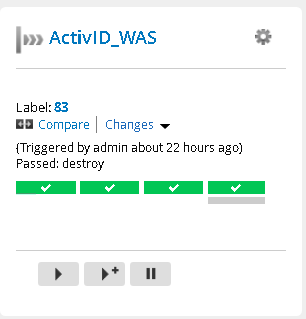


Here the ami-xxxxx got created & it has been updated in the consul as well. This image will be fetched and the changes will be applied during the terraform activity.

# 7 Terraform Activity – Provisioner

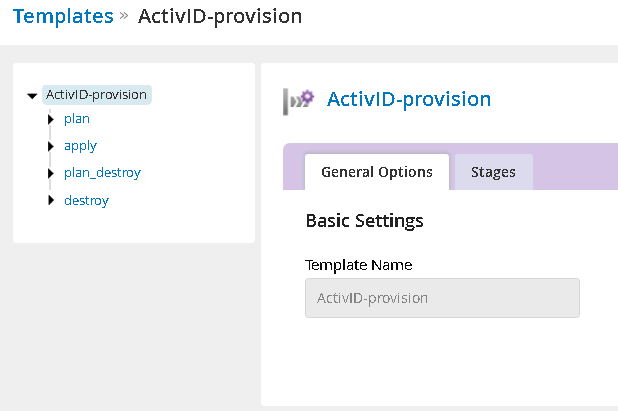
* + Note down the image ID (ami-xxxxxx) and proceed with the terraform activity.
  + Clone the terraform pipeline of HID RelA and name it as HID\_RELB.

Below is the pipeline we are using for terraform activity of RelB.



Create a new template for RelB and configure the stages & jobs similar to RelA.

* + - * Note:- Don’t change the order of the task execution.
    - Here we have created a template “ActivID-provision” and have added the 4 stages (plan, apply, plan-destroy & destroy).



* + Customize the artifacts & materials in the RelB pipeline w.r to 172.31.30.250 environment where AWS Gogs, GoCD and Nexus are running.
* Create the user data scripts which has to be executed as a part of terraform activity, here in RelB we have used 4 userdata script files:
  + - 1. Update-database-hostname.tpl
      2. Update-host\_entry\_data-init.tpl
      3. Activid-was-puppet-exports-init.tpl
      4. Banking-apps-puppet-run-init.tpl
* Place these scripts in the respective branch (r2-banking\_apps) and declare these variables in the terraform codes compute.tf & variables.tf. Below variables are being used for terraform activity for RelB

data\_port = "${var.data\_port}"

data\_name = "${var.data\_name}"

data\_hostname = "${var.data\_hostname}"

data\_port = "${var.data\_port}"

dom\_name = "${var.dom\_name}"

dom\_pwd = "${dom\_pwd}"

keyname = "${keyname}"

volume\_size = "${volume\_size}"

instance\_name = "${instance\_name}"

private\_ip = "${private\_ip}"

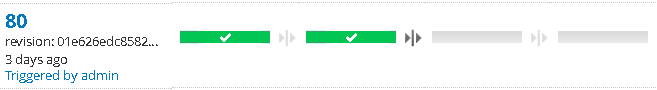
instance\_type = "${instance\_type}"

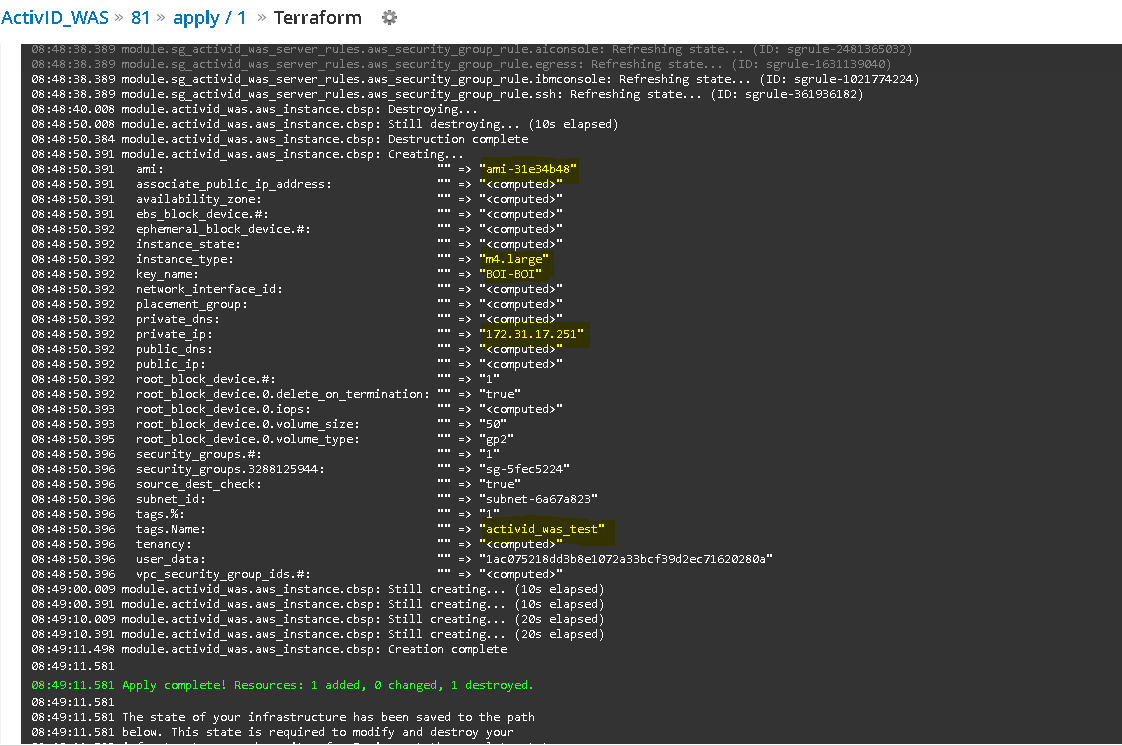
source\_cidr = "$source\_cidr}"

# 8 Terraform Activity – Provisioner (Run Pipeline)

* Once the above configuration has been completed, trigger the pipeline and wait for PLAN & APPLY stage completion.

Below is the output of a successful run.





* Once it is completed successfully, login to the IP address which is listed in the output and do the post validation checks.

**For ActivID:**

https://<ip-address>:9043/ibm/console 🡪 WAS console

https://<ip-address>:8445/aiconsole 🡪 HID console

# Document Version

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Changes** | **Documented By** | **Reviewed By** | **Approved By** |
| 0.2 | 21.11.2017 | Updated Version | Prabhu J |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2017. A global leader in consulting, technology and outsourcing services, the Group reported 2016 revenues of EUR 12.5 billion.

Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business ExperienceTM , and draws on Rightshore® its worldwide delivery model.

Learn more about us at [www.capgemini.com](http://www.capgemini.com)

*Rightshore® is a trademark belonging to Capgemini*

|  |  |
| --- | --- |
| Locations_Map_2013.png | Name Name  Function  Function  Phone: 00 00 00 00 00  email@capgemini.com |
| Name Name  Function  Function  Phone: 00 00 00 00 00  email@capgemini.com |
| Name Name  Function  Function  Phone: 00 00 00 00 00  email@capgemini.com |
| Name Name  Function  Function  Phone: 00 00 00 00 00  email@capgemini.com |
| www.capgemini.com  Description: C:\Users\UserSim\Desktop\DS_icons\128x128 shadows\facebook.png Description: C:\Users\UserSim\Desktop\DS_icons\128x128 shadows\linkedin.png Description: C:\Users\UserSim\Desktop\DS_icons\128x128 shadows\twitter.png Description: C:\Users\UserSim\Desktop\DS_icons\128x128 shadows\youtube.pngPicto_Slideshare.gif  The information contained in this presentation is proprietary. © 2017 Capgemini. All rights reserved. Rightshore® is a trademark belonging to Capgemini. | |