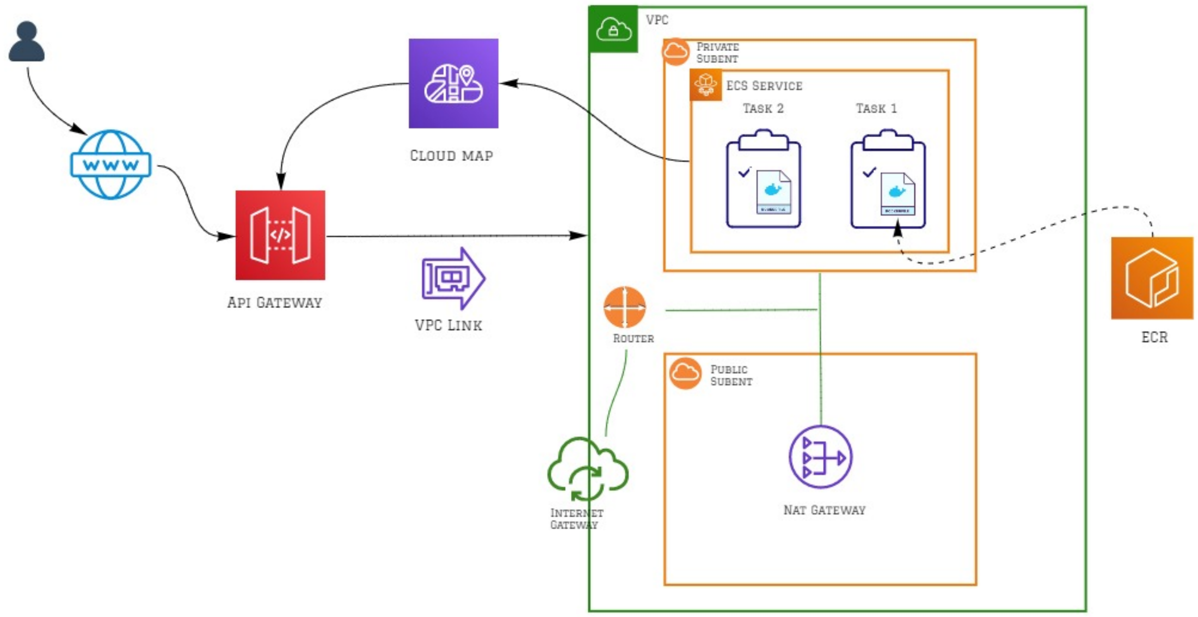
DevOps Coding Test

Table of Contents

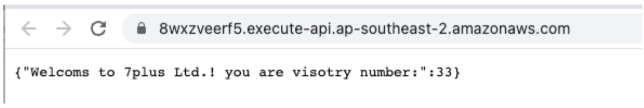
Architecture Overview



Folder Structure

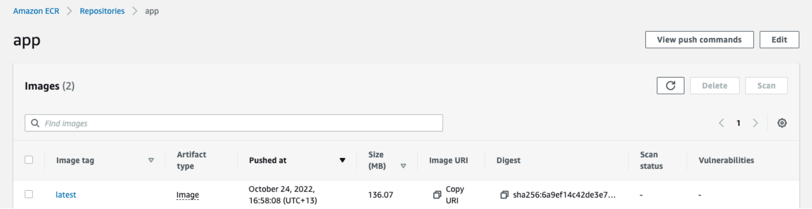
```
fargate-api-app
├── infra
│    ├── tf
│    ├── tf_modules
│    ├── roles
│    ├── sentinel
│    └── pieline.yml
└── Backend
    ├── Dockerfile
    ├── index.js
    └── package.json
```

- Backend:
  - A simple dockerfile that has only one stage based on bitnami node.js 14 image. It just copies the application source and runs npm install.
  - Index.js just displays hello world message with a random number:



```
{"Welcoms to 7plus Ltd.! you are visotry number:":33}
```

- The Image created by the dockerfile is pushed to an ecr repo. The Image URI and Image Name are provided to fargatecluster to create a service in the cluster.



app

Images (2)

☐	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
☐	latest	Image	October 24, 2022, 16:58:08 (UTC+13)	136.07	Copy URI	sha256:6a9ef14c42de3e7...	-	-

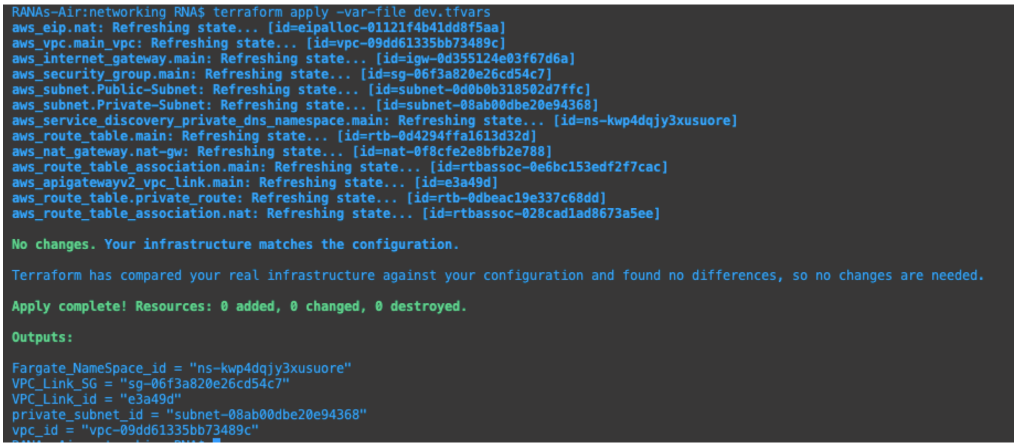
- infra/tf_modules

This folder contains two terraform modules:

1. networking module

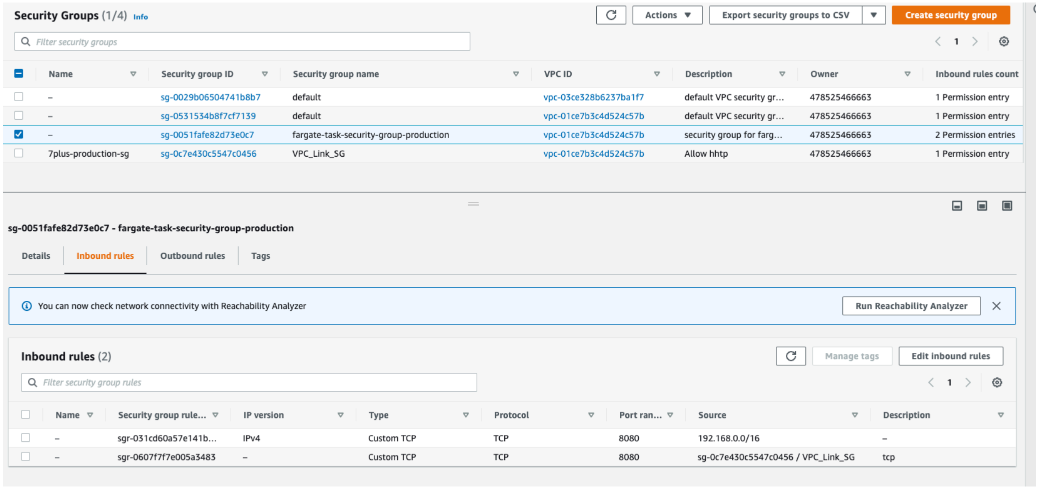
- It basically creates the networking infrastructure for the cluster.
- Creates 1 public subnets, and 1 private subnet, internet gateway and nat Gateway
- It creates a VPC Link: this aws service is managed by api gatewway. So api gateway can talk privately to the assigned resources in the VPC (the ECS container-based app). VPC Link
- It creates an AWS Cloud Map: this is for service discovery and its enabled in the ecs service.
- It creates a security group for the VPC Link.

Module Output:



```
RANAs-Air:networking RNA$ terraform apply -var-file dev.tfvars
aws_eip.nat: Refreshing state... [id=eipalloc-01121f4b41dd8f5aa]
aws_vpc.main_vpc: Refreshing state... [id=vpc-09dd61335bb73489c]
aws_internet_gateway.main: Refreshing state... [id=igw-0d355124e03f67d6a]
aws_security_group.main: Refreshing state... [id=sg-06f3a820e26cd54c7]
aws_subnet.Public-Subnet: Refreshing state... [id=subnet-0d0b0b318502d7ffc]
aws_subnet.Private-Subnet: Refreshing state... [id=subnet-08ab00dbe20e94368]
aws_service_discovery_private_dns_namespace.main: Refreshing state... [id=ns-kwp4dqjy3xusuore]
aws_route_table.main: Refreshing state... [id=rtb-0d4294ffa1613d32d]
aws_nat_gateway.nat-gw: Refreshing state... [id=nat-0f8cfe2e8bfb2e788]
aws_route_table_association.main: Refreshing state... [id=rtbassoc-0e6bc153edf2f7cac]
aws_apigatewayv2_vpc_link.main: Refreshing state... [id=e3a49d]
aws_route_table.private_route: Refreshing state... [id=rtb-0dbeac19e337c68dd]
aws_route_table_association.nat: Refreshing state... [id=rtbassoc-028cad1ad8673a5ee]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

Fargate_NameSpace_id = "ns-kwp4dqjy3xusuore"
VPC_Link_SG = "sg-06f3a820e26cd54c7"
VPC_Link_id = "e3a49d"
private_subnet_id = "subnet-08ab00dbe20e94368"
vpc_id = "vpc-09dd61335bb73489c"
```

- Vpc_id & private_subnet_id: private subnet for the ECS Cluster.



Security Groups (1/4)

☐	Name	Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules count
☐	–	sg-0029b06504741b8b7	default	vpc-03ce328b6237ba1f7	default VPC security gr...	478525466663	1 Permission entry
☐	–	sg-0531534b8f7cf7139	default	vpc-01ce7b3c4d524c57b	default VPC security gr...	478525466663	1 Permission entry
☑	–	sg-0051fafe82d73e0c7	fargate-task-security-group-production	vpc-01ce7b3c4d524c57b	security group for farg...	478525466663	2 Permission entries
☐	7plus-production-sg	sg-0c7e430c5547c0456	VPC_Link_SG	vpc-01ce7b3c4d524c57b	Allow hhtp	478525466663	1 Permission entry

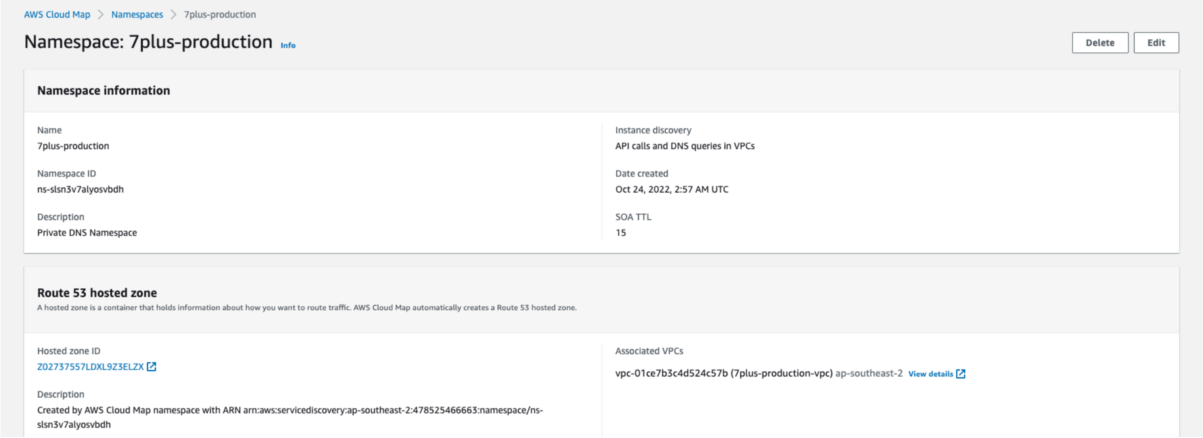
sg-0051fafe82d73e0c7 - fargate-task-security-group-production

You can now check network connectivity with Reachability Analyzer

Inbound rules (2)

☐	Name	Security group rule...	IP version	Type	Protocol	Port ran...	Source	Description
☐	–	sgr-031cd60a57e141b...	IPv4	Custom TCP	TCP	8080	192.168.0.0/16	–
☐	–	sgr-0607f7f7e005a3483	–	Custom TCP	TCP	8080	sg-0c7e430c5547c0456 / VPC_Link_SG	tcp

- VPC_Link_SG: We will need this for allowing tcp traffic in faregate task security group:
- fargate_namespace_id: for creating a service in cloudmap



Namespace: 7plus-production

Namespace information

Name
7plus-production

Namespace ID
ns-slsn3v7alyosvbdh

Description
Private DNS Namespace

Instance discovery
API calls and DNS queries in VPCs

Date created
Oct 24, 2022, 2:57 AM UTC

SOA TTL
15

Route 53 hosted zone
A hosted zone is a container that holds information about how you want to route traffic. AWS Cloud Map automatically creates a Route 53 hosted zone.

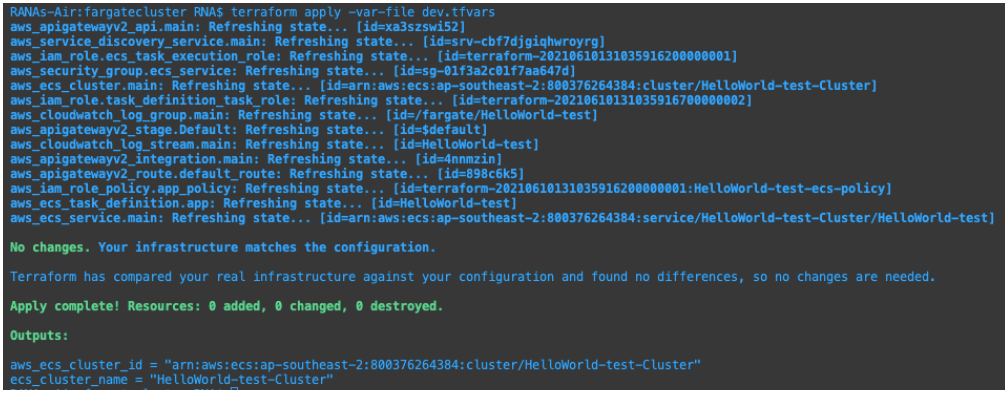
Hosted zone ID
Z02737557LDXL9Z3ELZX

Description
Created by AWS Cloud Map namespace with ARN arn:aws:servicediscovery:ap-southeast-2:478525466663:namespace/ns-slsn3v7alyosvbdh

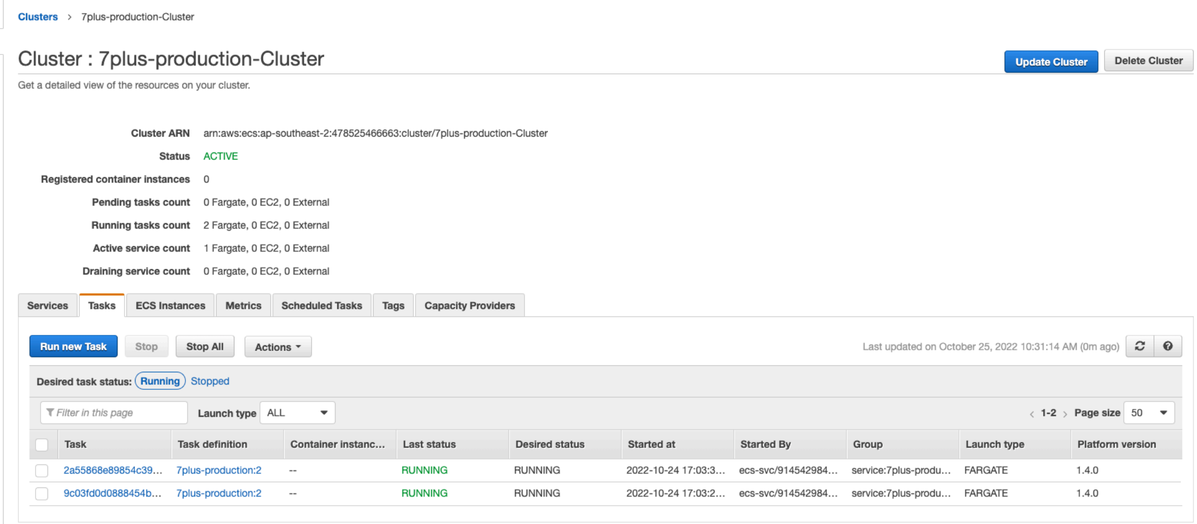
Associated VPCs
vpc-01ce7b3c4d524c57b (7plus-production-vpc) ap-southeast-2

2. fargate_cluster module

- It creates a ECS cluster, fargate ECS service, two tasks:



```
RANAs-Air:fargatecluster RNA$ terraform apply -var-file dev.tfvars
aws_apigatewayv2_api.main: Refreshing state... [id=xa3szswi52]
aws_service_discovery_service.main: Refreshing state... [id=srv-cbf7djgiqhwroyrg]
aws_iam_role.ecs_task_execution_role: Refreshing state... [id=terraform-20210610131035916200000001]
aws_security_group.ecs_service: Refreshing state... [id=sg-01f3a2c01f7aa647d]
aws_ecs_cluster.main: Refreshing state... [id=arn:aws:ecs:ap-southeast-2:800376264384:cluster/HelloWorld-test-Cluster]
aws_iam_role.task_definition_task_role: Refreshing state... [id=terraform-20210610131035916700000002]
aws_cloudwatch_log_group.main: Refreshing state... [id=/fargate/HelloWorld-test]
aws_apigatewayv2_stage.Default: Refreshing state... [id=$default]
aws_cloudwatch_log_stream.main: Refreshing state... [id=HelloWorld-test]
aws_apigatewayv2_integration.main: Refreshing state... [id=4nnmzin]
aws_apigatewayv2_route.default_route: Refreshing state... [id=898c6k5]
aws_iam_role_policy.app_policy: Refreshing state... [id=terraform-20210610131035916200000001:HelloWorld-test-ecs-policy]
aws_ecs_task_definition.app: Refreshing state... [id=HelloWorld-test]
aws_ecs_service.main: Refreshing state... [id=arn:aws:ecs:ap-southeast-2:800376264384:service/HelloWorld-test-Cluster/HelloWorld-test]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

aws_ecs_cluster_id = "arn:aws:ecs:ap-southeast-2:800376264384:cluster/HelloWorld-test-Cluster"
ecs_cluster_name = "HelloWorld-test-Cluster"
```



Cluster : 7plus-production-Cluster

Get a detailed view of the resources on your cluster.

Cluster ARN	arn:aws:ecs:ap-southeast-2:478525466663:cluster/7plus-production-Cluster
Status	ACTIVE
Registered container instances	0
Pending tasks count	0 Fargate, 0 EC2, 0 External
Running tasks count	2 Fargate, 0 EC2, 0 External
Active service count	1 Fargate, 0 EC2, 0 External
Draining service count	0 Fargate, 0 EC2, 0 External

Last updated on October 25, 2022 10:31:14 AM (0m ago)

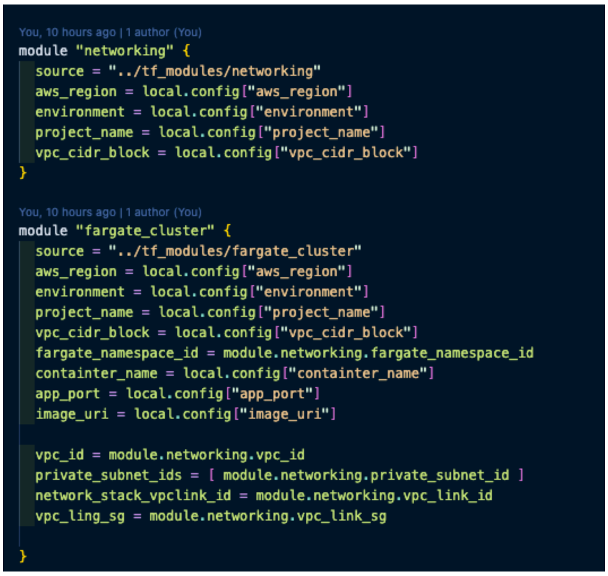
Desired task status: Running Stopped

Launch type ALL

1-2 Page size 50

□	Task	Task definition	Container instanc...	Last status	Desired status	Started at	Started By	Group	Launch type	Platform version
□	2a55868e89854c39...	7plus-production:2	--	RUNNING	RUNNING	2022-10-24 17:03:3...	ecs-svc/914542984...	service:7plus-produ...	FARGATE	1.4.0
□	9c03fd0d0888454b...	7plus-production:2	--	RUNNING	RUNNING	2022-10-24 17:03:2...	ecs-svc/914542984...	service:7plus-produ...	FARGATE	1.4.0

These two modules are used in infra/tf/main.tf. The output of the networking modules is used in faragate_clster module.



```
You, 10 hours ago | 1 author (You)
module "networking" {
  source = "../tf_modules/networking"
  aws_region = local.config["aws_region"]
  environment = local.config["environment"]
  project_name = local.config["project_name"]
  vpc_cidr_block = local.config["vpc_cidr_block"]
}

You, 10 hours ago | 1 author (You)
module "fargate_cluster" {
  source = "../tf_modules/fargate_cluster"
  aws_region = local.config["aws_region"]
  environment = local.config["environment"]
  project_name = local.config["project_name"]
  vpc_cidr_block = local.config["vpc_cidr_block"]
  fargate_namespace_id = module.networking.fargate_namespace_id
  containter_name = local.config["containter_name"]
  app_port = local.config["app_port"]
  image_uri = local.config["image_uri"]

  vpc_id = module.networking.vpc_id
  private_subnet_ids = [ module.networking.private_subnet_id ]
  network_stack_vpclink_id = module.networking.vpc_link_id
  vpc_ling_sg = module.networking.vpc_link_sg

}
```

Deployment

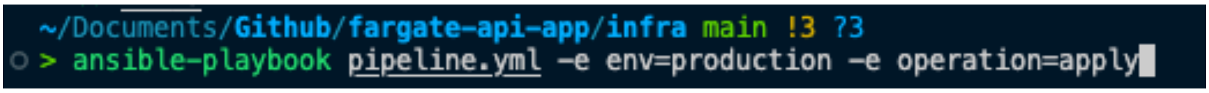
Terraform Backend Configuration

Initially, I have used s3 backend. By deploying tf/tf_backend that creates s3 bucked aws_dynamodb_table.

However, to use sentinel tests I have switched to terraform cloud. Hence, I have disabled s3-backend tasks in ansible role: infra/roles/tf_automation/tf-tasks-local.disabled and infra/tf/main_s3_backend.disabled

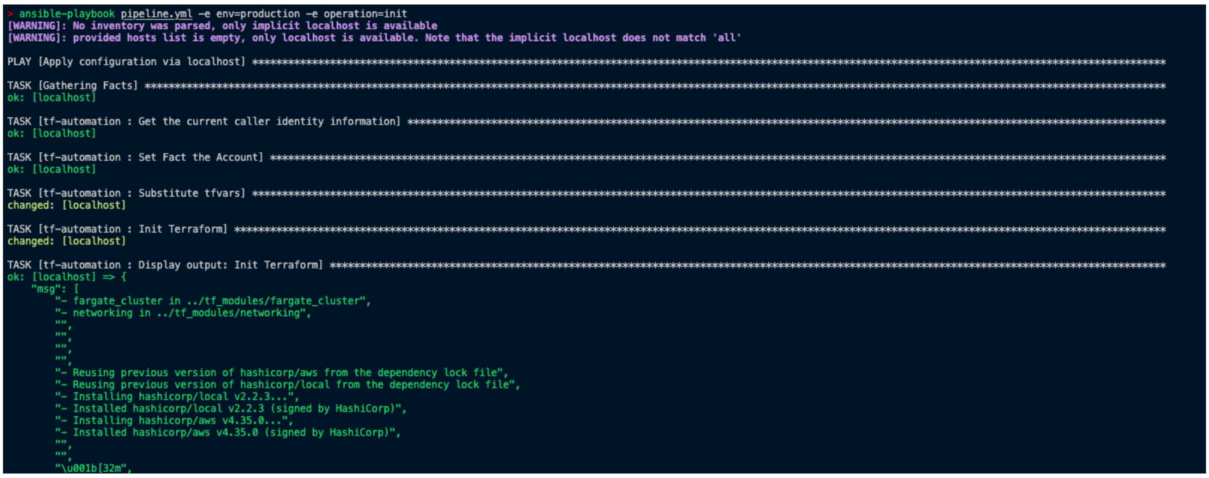
Managing Terraform Resources

I have created an ansible role infra/roles/tf_automation that replicate terraform commands: init, plan, apply and destory:



```
 ~/Documents/Github/fargate-api-app/infra main !3 ?3
○ > ansible-playbook pipeline.yml -e env=production -e operation=apply
```

The playbook is run from infra folder:

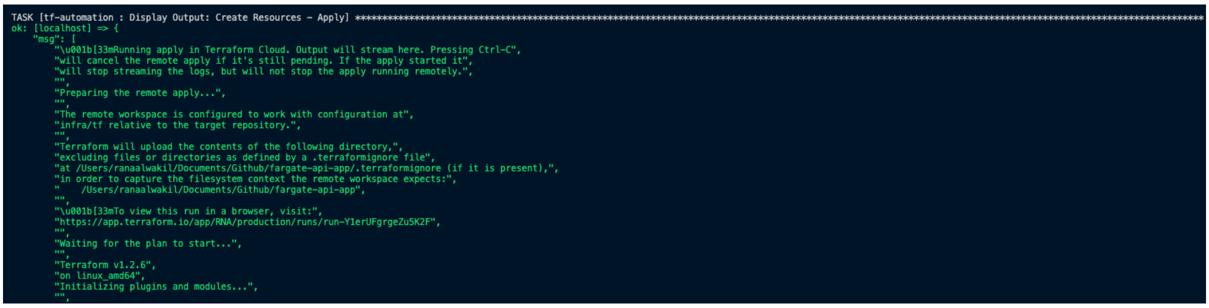


```
> ansible-playbook pipeline.yml -e env=production -e operation=init
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Apply configuration via localhost] ************************************************************************************************************

TASK [Gathering Facts] ******************************************************************************************************************************
ok: [localhost]

TASK [tf-automation : Get the current caller identity information] **********************************************************************************
ok: [localhost]

TASK [tf-automation : Set Fact the Account] *********************************************************************************************************
ok: [localhost]

TASK [tf-automation : Substitute tfvars] ************************************************************************************************************
changed: [localhost]

TASK [tf-automation : Init Terraform] ***************************************************************************************************************
changed: [localhost]

TASK [tf-automation : Display output: Init Terraform] ***********************************************************************************************
ok: [localhost] => {
    "msg": [
        "- fargate_cluster in ../tf_modules/fargate_cluster",
        "- networking in ../tf_modules/networking",
        "",
        "",
        "",
        "",
        "- Reusing previous version of hashicorp/aws from the dependency lock file",
        "- Reusing previous version of hashicorp/local from the dependency lock file",
        "- Installing hashicorp/local v2.2.3...",
        "- Installed hashicorp/local v2.2.3 (signed by HashiCorp)",
        "- Installing hashicorp/aws v4.35.0...",
        "- Installed hashicorp/aws v4.35.0 (signed by HashiCorp)",
        "",
        "",
        "\u001b[32m",
```

I have used

```
msg: "{{ (init.stdout | regex_replace('\\u001b.*0m', '')|trim).split('\n') }}"
```

In ansible debug tasks to produce a cleaner terraform output



```
TASK [tf-automation : Display Output: Create Resources - Apply] ***************************************************************************************************************************************************************
ok: [localhost] => {
    "msg": [
        "\u001b[33mRunning apply in Terraform Cloud. Output will stream here. Pressing Ctrl-C",
        "will cancel the remote apply if it's still pending. If the apply started it",
        "will stop streaming the logs, but will not stop the apply running remotely.",
        "",
        "Preparing the remote apply...",
        "",
        "The remote workspace is configured to work with configuration at",
        "infra/tf relative to the target repository.",
        "",
        "Terraform will upload the contents of the following directory,",
        "excluding files or directories as defined by a .terraformignore file",
        "at /Users/ranaalwakil/Documents/Github/fargate-api-app/.terraformignore (if it is present),",
        "in order to capture the filesystem context the remote workspace expects:",
        "    /Users/ranaalwakil/Documents/Github/fargate-api-app",
        "",
        "\u001b[33mTo view this run in a browser, visit:",
        "https://app.terraform.io/app/RNA/production/runs/run-Y1erUFgrgeZu5K2F",
        "",
        "Waiting for the plan to start...",
        "",
        "Terraform v1.2.6",
        "on linux_amd64",
        "Initializing plugins and modules...",
        "",
```

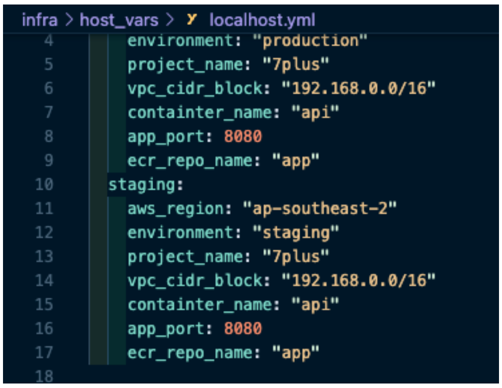
Note: when using s3 backend, terraform plan can be saved with:

```
terraform plan -out=plan.tfplan;
```

and then can be used later in ansible task to create the resources (please check infra/roles/tf_automation/tf-tasks-local.disabled). Although its best practice, unfortunately, this option isn't available for terraform cloud.

Variables Passing

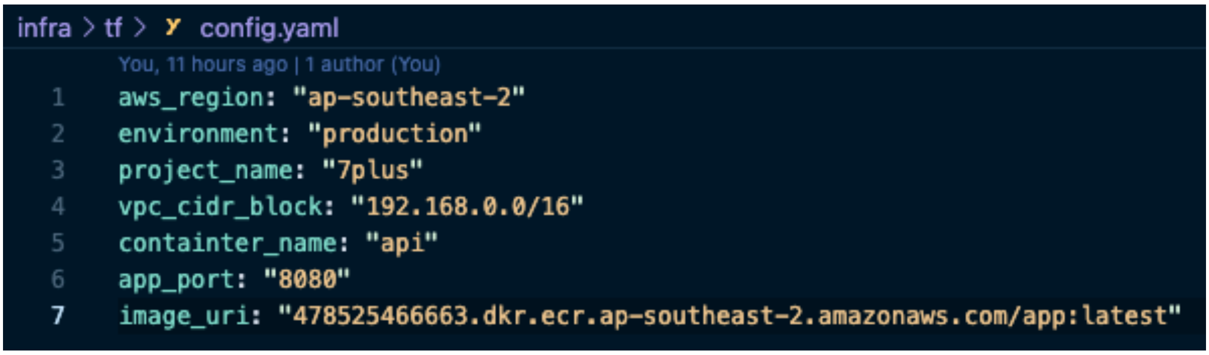
Variable are initialised in infra/host_vars/localhost.yml



infra > host_vars > localhost.yml

```
    environment: "production"
    project_name: "7plus"
    vpc_cidr_block: "192.168.0.0/16"
    containter_name: "api"
    app_port: 8080
    ecr_repo_name: "app"
  staging:
    aws_region: "ap-southeast-2"
    environment: "staging"
    project_name: "7plus"
    vpc_cidr_block: "192.168.0.0/16"
    containter_name: "api"
    app_port: 8080
    ecr_repo_name: "app"

```

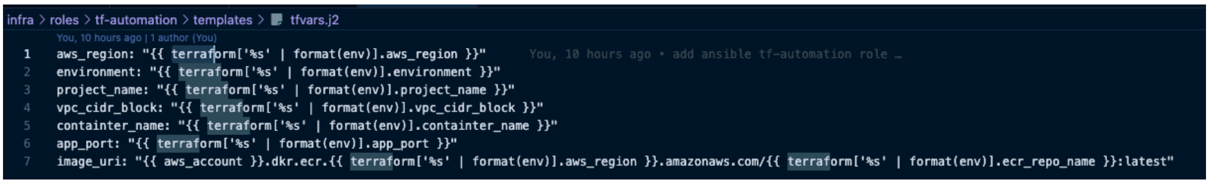
Its then parsed in ansbible role tf_automation using a jinja2 template tfvars.j2 to create config.yml
Note: I have created a more 'complex' variable passing scenario: image_uri variable uses ansible fact aws_account defined in tf_automation tasks



infra > tf > config.yaml

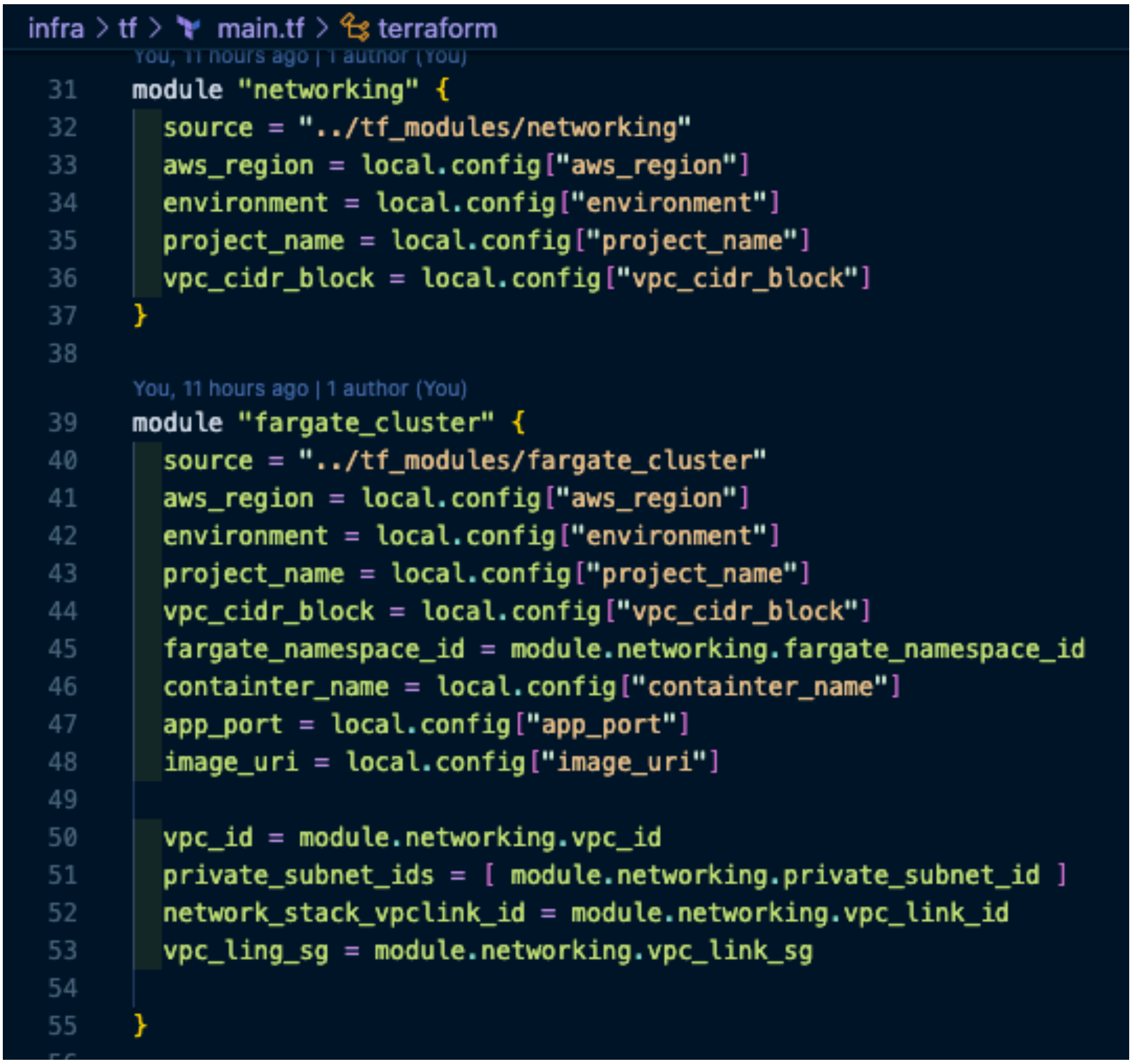
You, 11 hours ago | 1 author (You)

```
aws_region: "ap-southeast-2"
environment: "production"
project_name: "7plus"
vpc_cidr_block: "192.168.0.0/16"
containter_name: "api"
app_port: "8080"
image_uri: "478525466663.dkr.ecr.ap-southeast-2.amazonaws.com/app:latest"
```


infra > roles > tf-automation > templates > tfvars.j2
You, 10 hours ago | 1 author (You)
You, 10 hours ago • add ansible tf-automation role …

```
aws_region: "{{ terraform['%s' | format(env)].aws_region }}"
environment: "{{ terraform['%s' | format(env)].environment }}"
project_name: "{{ terraform['%s' | format(env)].project_name }}"
vpc_cidr_block: "{{ terraform['%s' | format(env)].vpc_cidr_block }}"
containter_name: "{{ terraform['%s' | format(env)].containter_name }}"
app_port: "{{ terraform['%s' | format(env)].app_port }}"
image_uri: "{{ aws_account }}.dkr.ecr.{{ terraform['%s' | format(env)].aws_region }}.amazonaws.com/{{ terraform['%s' | format(env)].ecr_repo_name }}:latest"
```

The config.yaml file is used in main.tf. That way we don't need to define variable.tf in the root module and update it everytime we add a new variable.


infra > tf > main.tf > terraform
You, 11 hours ago | 1 author (You)
You, 11 hours ago | 1 author (You)

```
module "networking" {
  source = "../tf_modules/networking"
  aws_region = local.config["aws_region"]
  environment = local.config["environment"]
  project_name = local.config["project_name"]
  vpc_cidr_block = local.config["vpc_cidr_block"]
}

module "fargate_cluster" {
  source = "../tf_modules/fargate_cluster"
  aws_region = local.config["aws_region"]
  environment = local.config["environment"]
  project_name = local.config["project_name"]
  vpc_cidr_block = local.config["vpc_cidr_block"]
  fargate_namespace_id = module.networking.fargate_namespace_id
  containter_name = local.config["containter_name"]
  app_port = local.config["app_port"]
  image_uri = local.config["image_uri"]

  vpc_id = module.networking.vpc_id
  private_subnet_ids = [ module.networking.private_subnet_id ]
  network_stack_vpclink_id = module.networking.vpc_link_id
  vpc_ling_sg = module.networking.vpc_link_sg

}
```

Testing

I have used sentinel and created 3 test cases:

- enforce-ssh-disabled

To check for any security group that have SSH open to CIDR "0.0.0.0/0" for ingress rules.

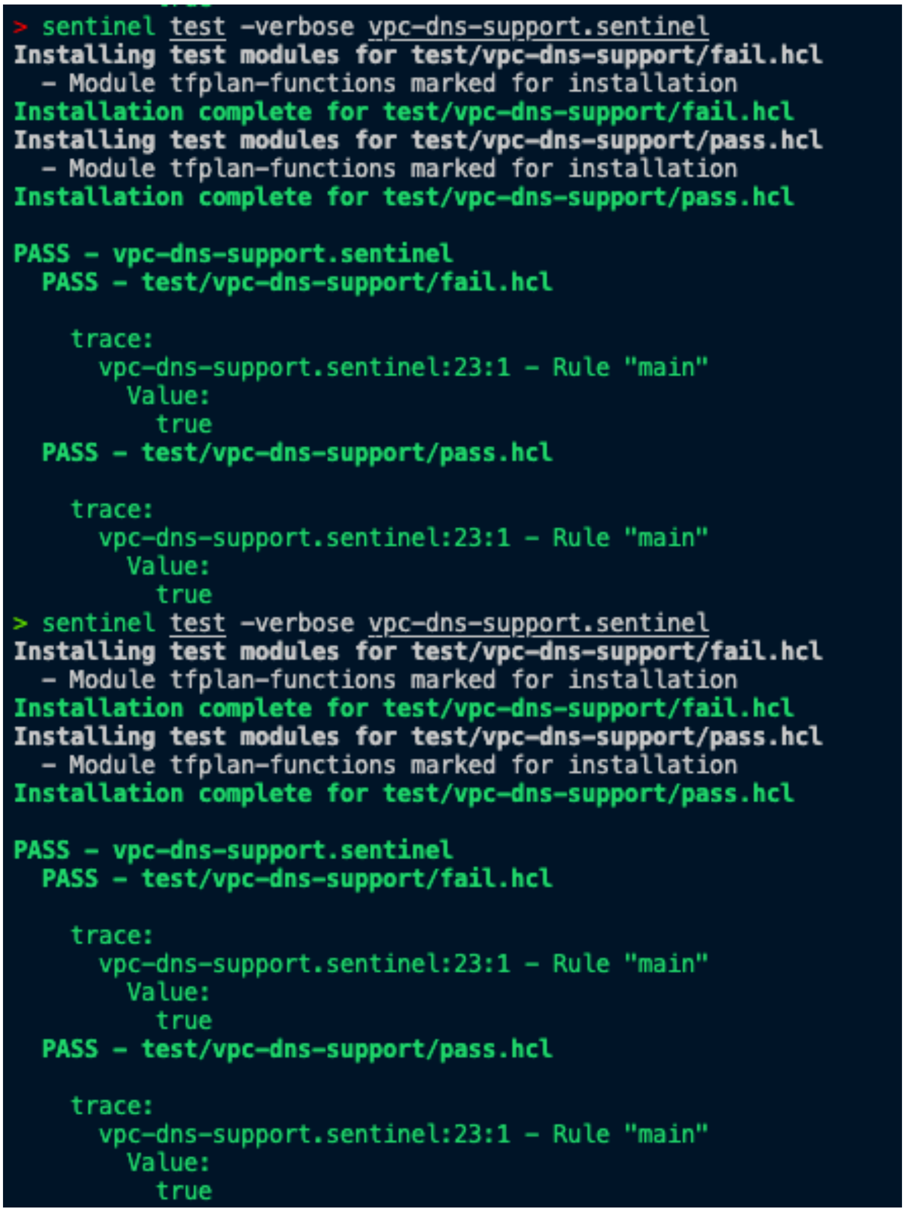
- limit-cost-and-percentage-increase

This policy restricts both the total monthly cost and the percentage increase in the monthly cost that would be incurred if the current plan were applied

- vpc-dns-support

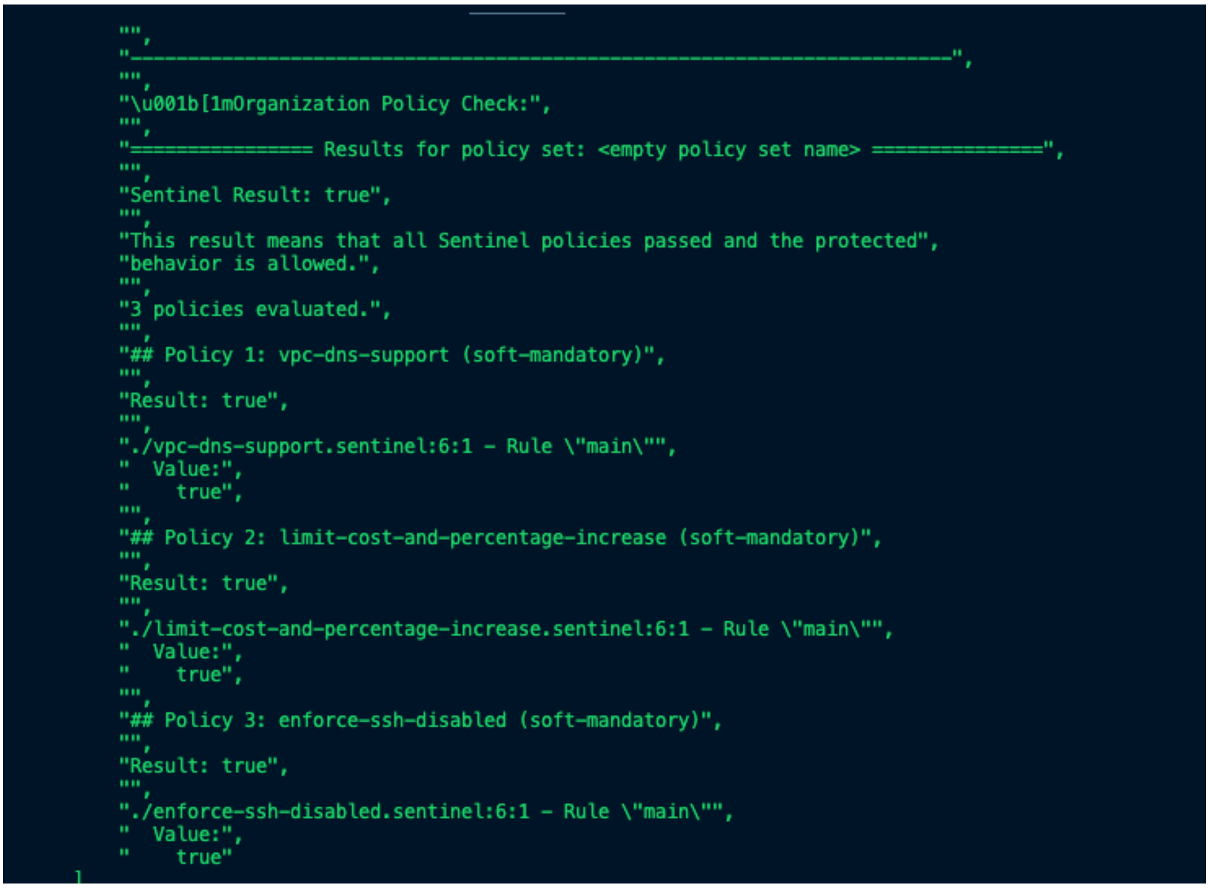
This policy checks that the VPC supports DNS

I have downloaded the mock data from terraform cloud and test these policies locally:



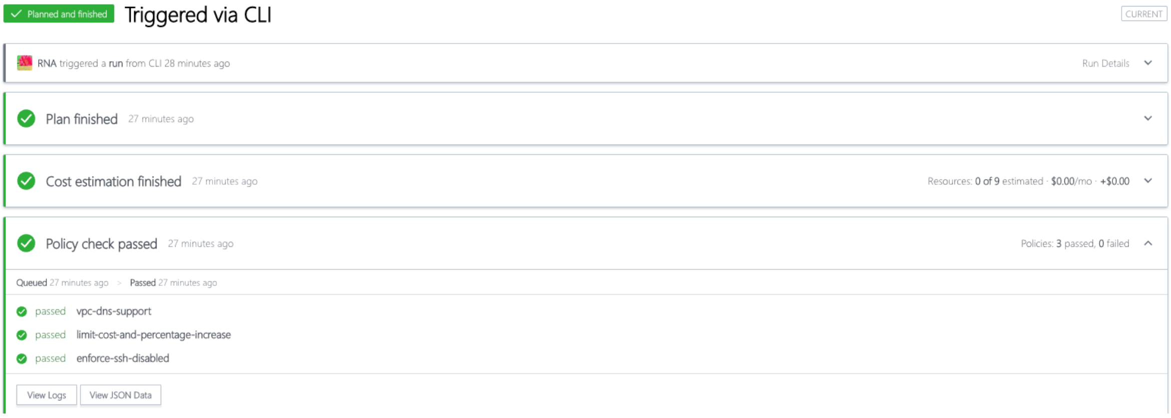
```
> sentinel test -verbose vpc-dns-support.sentinel
Installing test modules for test/vpc-dns-support/fail.hcl
  - Module tfplan-functions marked for installation
Installation complete for test/vpc-dns-support/fail.hcl
Installing test modules for test/vpc-dns-support/pass.hcl
  - Module tfplan-functions marked for installation
Installation complete for test/vpc-dns-support/pass.hcl

PASS - vpc-dns-support.sentinel
  PASS - test/vpc-dns-support/fail.hcl

    trace:
      vpc-dns-support.sentinel:23:1 - Rule "main"
        Value:
          true
  PASS - test/vpc-dns-support/pass.hcl

    trace:
      vpc-dns-support.sentinel:23:1 - Rule "main"
        Value:
          true
> sentinel test -verbose vpc-dns-support.sentinel
Installing test modules for test/vpc-dns-support/fail.hcl
  - Module tfplan-functions marked for installation
Installation complete for test/vpc-dns-support/fail.hcl
Installing test modules for test/vpc-dns-support/pass.hcl
  - Module tfplan-functions marked for installation
Installation complete for test/vpc-dns-support/pass.hcl

PASS - vpc-dns-support.sentinel
  PASS - test/vpc-dns-support/fail.hcl

    trace:
      vpc-dns-support.sentinel:23:1 - Rule "main"
        Value:
          true
  PASS - test/vpc-dns-support/pass.hcl

    trace:
      vpc-dns-support.sentinel:23:1 - Rule "main"
        Value:
          true
```

Then created a policy set in terraform cloud to run these policies before terraform apply:



```
          "",
          "------------------------------------------------------------------------",
          "",
          "\u001b[1mOrganization Policy Check:",
          "",
          "================ Results for policy set: <empty policy set name> ===============",
          "",
          "Sentinel Result: true",
          "",
          "This result means that all Sentinel policies passed and the protected",
          "behavior is allowed.",
          "",
          "3 policies evaluated.",
          "",
          "## Policy 1: vpc-dns-support (soft-mandatory)",
          "",
          "Result: true",
          "",
          "./vpc-dns-support.sentinel:6:1 - Rule \"main\"",
          "  Value:",
          "    true",
          "",
          "## Policy 2: limit-cost-and-percentage-increase (soft-mandatory)",
          "",
          "Result: true",
          "",
          "./limit-cost-and-percentage-increase.sentinel:6:1 - Rule \"main\"",
          "  Value:",
          "    true",
          "",
          "## Policy 3: enforce-ssh-disabled (soft-mandatory)",
          "",
          "Result: true",
          "",
          "./enforce-ssh-disabled.sentinel:6:1 - Rule \"main\"",
          "  Value:",
          "    true"
      ]
```

Output from terraform UI:



✓ Planned and finished **Triggered via CLI**

RNA triggered a **run** from CLI 28 minutes ago

✓ Plan finished 27 minutes ago

✓ Cost estimation finished 27 minutes ago — Resources: 0 of 9 estimated · $0.00/mo · +$0.00

✓ Policy check passed 27 minutes ago — Policies: 3 passed, 0 failed

Queued 27 minutes ago > Passed 27 minutes ago

✓ passed vpc-dns-support

✓ passed limit-cost-and-percentage-increase

✓ passed enforce-ssh-disabled

Enhancements

- Design first. It's a lot cheaper to invest on spec first and iterate on it. Should we decide from the beginning to divide the backend and frontend component? It really depends on the teams that are working on the project. But it's important to distinguish these components from build and deploy point of view.
- Using git: I haven't really used git "effectively". I would like to make more smaller commits as it will make it easier to revert code to a previous state and write meaningful commits.
- For the IAM roles: use a resource-based security policies. And using a role for ansible deployments.
- For the docker file: We used the same file for test and prod. In a more complex application, its recommended to add another step to create a minimal Docker image that only consists of the application source, modules and Node.js runtime.