

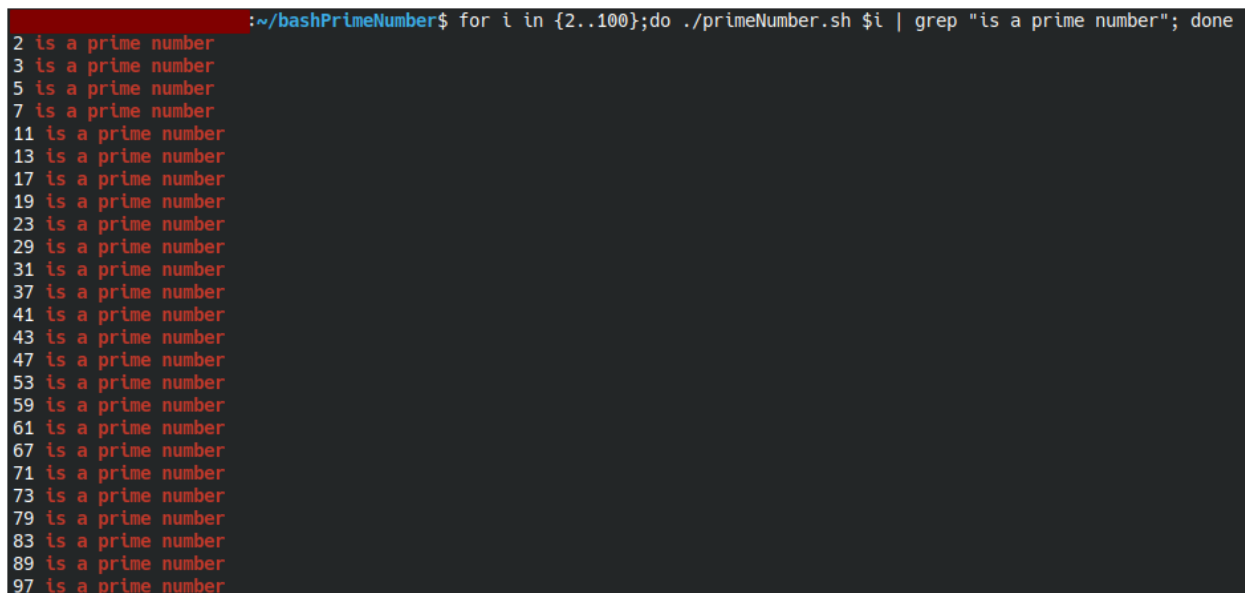
1. Buatlah bash script yang bisa mengecek apakah suatu bilangan itu bilangan prima atau bukan

```
#!/bin/bash
```

```
#argument $1 is the inputted number
number=$1
i=2
while [ $i -lt $number ]
do
    if [[ $(( $number % $i )) -eq 0 ]]
    then
        echo "$number is not a prime number"
        exit 1
    fi
    ((i++))
done
echo "$number is a prime number"
```

Script di atas digunakan untuk mencari angka prima dengan memeriksa apakah angka itu tidak dapat dibagi dengan angka 2 sampai dengan angka itu sendiri yang dikurangi 1.

Hasil testnya adalah sebagai berikut:



```
~/bashPrimeNumber$ for i in {2..100};do ./primeNumber.sh $i | grep "is a prime number"; done
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
41 is a prime number
43 is a prime number
47 is a prime number
53 is a prime number
59 is a prime number
61 is a prime number
67 is a prime number
71 is a prime number
73 is a prime number
79 is a prime number
83 is a prime number
89 is a prime number
97 is a prime number
```

2. Deploy service demo microservice

<https://github.com/docker-archive/swarm-microservice-demo-v1> dengan spesifikasi environment sebagai berikut **(Point 80)** :

Untuk soal ini saya melakukan forking dari repositori di atas lalu menambahkan beberapa hal pada repositori forkingnya di <https://github.com/nizarakbarm/swarm-microservice-demo-v1>, yaitu:

- Terraform Infrastruktur, yang isi main.yaml-nya adalah seperti yang ada di <https://raw.githubusercontent.com/nizarakbarm/swarm-microservice-demo-v1/master/Terraform-Infrastructure/main.tf> kegunaan terraform tersebut adalah untuk membuat tiga vm lightsail dengan menggunakan blueprint-id, availability_zone_name, key_pair_name, dan bundle_id yang disediakan di variables.tf. Pada main.tf saya menggunakan count=3 untuk membuat 3 instance lightsail sekaligus.
 - Untuk deploy terraform ini perlu dilakukan *terraform init* untuk inisiasi package provider terraform lalu *terraform plan* untuk cek apakah settingannya sudah sesuai
 - Setelah itu perlu melakukan *terraform apply*

```

denrell ~ - ssh - microservice-demo-v1: Terraform-Infrastructure
└─ terraform plan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# aws_lightsail_instance.lightsail_k8s[0] will be created
+ resource "aws_lightsail_instance" "lightsail_k8s" {
  + arn                = (known after apply)
  + availability_zone   = "ap-southeast-1a"
  + blueprint_id       = "ubuntu_22_04"
  + bundle_id          = "large_3_8"
  + cpu_count          = (known after apply)
  + created_at         = (known after apply)
  + id                 = (known after apply)
  + ip_address_type    = "dualstack"
  + ipv6_addresses     = (known after apply)
  + is_static_ip       = (known after apply)
  + key_pair_name      = "amazon"
  + name               = "node-0"
  + private_ip_address = (known after apply)
  + public_ip_address  = (known after apply)
  + ram_size           = (known after apply)
  + tags               = {
    + "node" = "k8s"
  }
  + tags_all           = {
    + "node" = "k8s"
  }
  + username           = (known after apply)
}

# aws_lightsail_instance.lightsail_k8s[1] will be created
+ resource "aws_lightsail_instance" "lightsail_k8s" {
  + arn                = (known after apply)
  + availability_zone   = "ap-southeast-1a"
  + blueprint_id       = "ubuntu_22_04"
  + bundle_id          = "large_3_8"
  + cpu_count          = (known after apply)
  + created_at         = (known after apply)
  + id                 = (known after apply)
  + ip_address_type    = "dualstack"
  + ipv6_addresses     = (known after apply)
  + is_static_ip       = (known after apply)
  + key_pair_name      = "amazon"
  + name               = "node-1"
  + private_ip_address = (known after apply)
  + public_ip_address  = (known after apply)
  + ram_size           = (known after apply)
  + tags               = {
    + "node" = "k8s"
  }
  + tags_all           = {
    + "node" = "k8s"
  }
  + username           = (known after apply)
}

# aws_lightsail_instance.lightsail_k8s[2] will be created
+ resource "aws_lightsail_instance" "lightsail_k8s" {
  + arn                = (known after apply)
  + availability_zone   = "ap-southeast-1a"
  + blueprint_id       = "ubuntu_22_04"
  + bundle_id          = "large_3_8"
  + cpu_count          = (known after apply)
  + created_at         = (known after apply)
  + id                 = (known after apply)
  + ip_address_type    = "dualstack"
  + ipv6_addresses     = (known after apply)
  + is_static_ip       = (known after apply)
  + key_pair_name      = "amazon"
  + name               = "node-2"
  + private_ip_address = (known after apply)
  + public_ip_address  = (known after apply)
  + ram_size           = (known after apply)
  + tags               = {
    + "node" = "k8s"
  }
  + tags_all           = {
    + "node" = "k8s"
  }
  + username           = (known after apply)
}

Plan: 3 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

```

```
devnull ~/aws-microservice-demo-v1/terraform-infrastructure
$ terraform apply
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

create

Terraform will perform the following actions:

```
# aws Lightsail Instance lightsail_k8s[0] will be created
resource "aws_lightsail_instance" "lightsail_k8s" {
  arm = (known after apply)
  availability_zone = "ap-southeast-1a"
  blueprint_id = "ubuntu_22_04"
  bundle_id = "large_3_0"
  cpu_count = (known after apply)
  created_at = (known after apply)
  id = (known after apply)
  ip_address_type = "dualstack"
  ipv6_addresses = (known after apply)
  is_static_ip = (known after apply)
  key_pair_name = "amazon"
  name = "node-0"
  private_ip_address = (known after apply)
  public_ip_address = (known after apply)
  ram_size = (known after apply)
  tags = {
    "node" = "k8s"
  }
  tags_all = {
    "node" = "k8s"
  }
  username = (known after apply)
}

# aws Lightsail Instance lightsail_k8s[1] will be created
resource "aws_lightsail_instance" "lightsail_k8s" {
  arm = (known after apply)
  availability_zone = "ap-southeast-1a"
  blueprint_id = "ubuntu_22_04"
  bundle_id = "large_3_0"
  cpu_count = (known after apply)
  created_at = (known after apply)
  id = (known after apply)
  ip_address_type = "dualstack"
  ipv6_addresses = (known after apply)
  is_static_ip = (known after apply)
  key_pair_name = "amazon"
  name = "node-1"
  private_ip_address = (known after apply)
  public_ip_address = (known after apply)
  ram_size = (known after apply)
  tags = {
    "node" = "k8s"
  }
  tags_all = {
    "node" = "k8s"
  }
  username = (known after apply)
}

# aws Lightsail Instance lightsail_k8s[2] will be created
resource "aws_lightsail_instance" "lightsail_k8s" {
  arm = (known after apply)
  availability_zone = "ap-southeast-1a"
  blueprint_id = "ubuntu_22_04"
  bundle_id = "large_3_0"
  cpu_count = (known after apply)
  created_at = (known after apply)
  id = (known after apply)
  ip_address_type = "dualstack"
  ipv6_addresses = (known after apply)
  is_static_ip = (known after apply)
  key_pair_name = "amazon"
  name = "node-2"
  private_ip_address = (known after apply)
  public_ip_address = (known after apply)
  ram_size = (known after apply)
  tags = {
    "node" = "k8s"
  }
  tags_all = {
    "node" = "k8s"
  }
  username = (known after apply)
}
```

Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_lightsail_instance.lightsail_k8s[2]: Creating...

Enter a value: yes

```
aws_lightsail_instance.lightsail_k8s[2]: Creating...
aws_lightsail_instance.lightsail_k8s[0]: Creating...
aws_lightsail_instance.lightsail_k8s[1]: Creating...
aws_lightsail_instance.lightsail_k8s[0]: Still creating... [10s elapsed]
aws_lightsail_instance.lightsail_k8s[2]: Still creating... [10s elapsed]
aws_lightsail_instance.lightsail_k8s[1]: Still creating... [10s elapsed]
aws_lightsail_instance.lightsail_k8s[0]: Still creating... [20s elapsed]
aws_lightsail_instance.lightsail_k8s[2]: Still creating... [20s elapsed]
aws_lightsail_instance.lightsail_k8s[1]: Still creating... [20s elapsed]
aws_lightsail_instance.lightsail_k8s[1]: Creation complete after 28s [id=node-1]
aws_lightsail_instance.lightsail_k8s[2]: Creation complete after 28s [id=node-2]
aws_lightsail_instance.lightsail_k8s[0]: Creation complete after 28s [id=node-0]
```

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.

- Lalu saya membuat Terraform-RKE2 yang kegunaannya untuk deploy RKE2 di lightsail. Pada script <https://raw.githubusercontent.com/nizarakbarm/swarm-microservice-demo-v1/master/Terraform-Infrastructure/main.tf> saya membuat remote file provisioning untuk menyakin `rke2configshell.sh` yang isinya adalah script shell untuk deploy rke2. Tetapi sebelum dideploy perlu ubah beberapa nilai di `variables.tf` yaitu `[NODE0-IP]`, `[NODE1-IP]`, dan `[NODE2-IP]` untuk menggunakan ip dari masing-masing node lightsail. Selain itu perlu menambahkan path private key di `[PATH_SSH_PRIVATE_KEY]`. Lalu di `BashDeployRKE2` ada dua script yang digunakan yaitu:
 - `node0_RKE2.sh`: untuk deploy rke2 di node-0, di script ini perlu ubah `IPNODE0` menjadi ip dari node0
 - `node1_and_2_RKE2.sh`: untuk deploy rke2 di node-1 dan node-2, sama seperti sebelumnya perlu ubah `IPNODE0` menjadi ip dari node-0

Selain ubah beberapa settingan di terraform dan script, disarankan untuk menambahkan `[ip-node0] node0` di `/etc/hosts` dari masing masing node.

- Setelah dilakukan provision script, saya buat data source menggunakan provider `command` untuk membuat UUID dari hasil perintah `cat /proc/sys/kernel/random/uuid`.
- Nilai UUID yang diperoleh akan digunakan untuk mengubah `UUID_VAL` di script `/home/ubuntu/rke2configshell.sh` yang sebelumnya sudah diupload saat provisioning.
- Setelah itu semua script di node-0, node-1, dan node-2 diubah permissionnya menjadi 755
- Setelah diubah permissionnya dilakukan eksekusi script dengan urutannya adalah:
 - Eksekusi di node-0 dengan resource `terraform_data.configureRKE2_node_0`. Eksekusinya dilakukan di satu node terlebih dahulu karena dua node lainnya membutuhkan satu node untuk join ke cluster.
 - Selanjutnya baru dilakukan eksekusi di node 1 dan 2 dengan resource `terraform_data configureRKE2_node_1_2`.
- Untuk dua script bash tambahan yang digunakan dapat dicek di https://raw.githubusercontent.com/nizarakbarm/swarm-microservice-demo-v1/master/Terraform-RKE2/BashDeployRKE2/node0_RKE2.sh dan https://raw.githubusercontent.com/nizarakbarm/swarm-microservice-demo-v1/master/Terraform-RKE2/BashDeployRKE2/node1_and_2_RKE2.sh.
- Untuk implementasi terraform ini perlu dilakukan *terraform init* karena terdapat satu provider yaitu `command`. Terraform init digunakan untuk inisiasi package provider yang dibutuhkan.
- Setelah itu dilakukan *terraform plan* untuk memeriksa konfigurasi sebelum diaplikasikan
- Jika sudah yakin, dijalankan *terraform apply* untuk mengaplikasikan konfigurasinya.
- Apabila rke2-server sudah berjalan maka servicenya akan berjalan jika dicek di `systemctl status rke2-server`

Setelah rke2-server selesai, saya juga membuat *ssh local tunnel* menggunakan `ssh`

```
ubuntu@52.77.228.35 -i /home/devnull/.ssh/amazon -N -f -L
```

6443:52.77.228.35:**6443** agar dapat mengakses port 6443 dari lokal sehingga dapat dengan mudah diakses oleh jenkins.

IPv4 Firewall [?](#)

Create rules to open ports to the internet, or to a specific IPv4 address or range.

[Learn more about firewall rules](#)

+ Add rule

Specify a port and protocol to open. Specify a port range using a dash, such as 0 - 65535.

Application	Protocol	Port or range	<input checked="" type="checkbox"/> Restrict to IP address
Custom	TCP	6443	

Specify an IP address allowed to connect to your instance. You can specify a range of IP addresses using a dash, or using CIDR notation. [Learn more](#)

Source IP address (192.0.2.0) or range (192.0.2.0-192.0.2.255 or 192.0.2.0/24)

54.255.240.90	×
52.77.228.35	×
13.212.194.75	×

+ Add another

Cancel ☒ Create ☒

Duplicate rule for IPv6 ☐

Application	Protocol	Port or range / Code	Restricted to	
SSH	TCP	22	Any IPv4 address	

Selain itu saya juga membuat settingan allow firewall untuk port 6443, 9345, dan 10250 dengan batasan hanya dapat diakses oleh ipv4 dan ipv6 dari 3 node di lightsail yang saya buat. Untuk screenshot salah satu settingan firewallnya adalah seperti di atas.

CI

Untuk CI saya menggunakan jenkins, langkah-langkah awal yang saya lakukan adalah:

1. Membuat multibranch pipeline

Enter an item name

» Required field



Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.



Organization Folder

Creates a set of multibranch project subfolders by scanning for repositories.

OK

2. Setup Branch Sources dan credentials-nya. Di sini saya menggunakan kredensial dari github access token.

The screenshot shows the 'Branch Sources' configuration in Jenkins. It is for a GitHub repository. The 'Credentials' dropdown is set to 'my github credential'. The 'Repository HTTPS URL' is 'https://github.com/nizarakbarm/swarm-microservice-demo-v1.git'. There is a 'Validate' button. The 'Behaviors' section is expanded, showing 'Discover branches' with a strategy of 'Exclude branches that are also filed as PRs'.

The screenshot shows the 'Add Credentials' dialog box in Jenkins. The 'Domain' is 'Global credentials (unrestricted)'. The 'Kind' is 'Username with password'. The 'Scope' is 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' is 'nizarakbarm04@gmail.com'. The 'Treat username as secret' checkbox is checked. The 'Password' field is masked with dots. The 'ID' is 'github-credential'. The 'Description' is 'my github credential'. There are 'Add', 'Cancel', 'Save', and 'Apply' buttons at the bottom.

3. Selanjutnya saya deploy deployment PostgreSQL menggunakan local-file, deploymentnya menggunakan yaml berikut:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: postgresdb-persistent-volume
```

```
  labels:
    type: local
    app: postgresdb
spec:
  storageClassName: manual
  capacity:
    storage: 8Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  hostPath:
    path: "/data/db"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: db-persistent-volume-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 8Gi
---
apiVersion: v1
kind: Secret
metadata:
  name: postgres-secret
data:
  POSTGRES_PASSWORD: cGc4Njc1MzA5
stringData:
  POSTGRES_USER: postgres
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-configmap
  labels:
```



```
    app: postgresdb
data:
  POSTGRES_DB: postgres
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: store
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgresdb
  template:
    metadata:
      labels:
        app: postgresdb
    spec:
      containers:
      - name: postgresdb
        image: postgres:9.4.26
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m"
          limits:
            memory: "1Gi"
            cpu: "500m"
        ports:
          - containerPort: 5432
        envFrom:
          - secretRef:
              name: postgres-secret
          - configMapRef:
              name: postgres-configmap
        volumeMounts:
          - mountPath: /var/lib/postgres/data
            name: db-data
      volumes:
      - name: db-data
```

```
    persistentVolumeClaim:
      claimName: db-persistent-volume-claim

---
apiVersion: v1
kind: Service
metadata:
  name: store
spec:
  selector:
    app: postgresdb
  ports:
    - protocol: TCP
      port: 5432
```

Untuk postgresql ini juga saya test menggunakan pipeline dari jenkins. Hasilnya adalah:

- Status
- </>

Changes
- ▶

Build Now
- ⚙️

View Configuration
- 🔍

Full Stage View
- 🔗

GitHub
- ?

Pipeline Syntax

Build History trend

Filter builds...

✓ #9

Aug 27, 2023, 2:19 AM

✗ #8

Aug 27, 2023, 2:14 AM

✗ #7

Aug 27, 2023, 2:12 AM

✗ #6

Aug 27, 2023, 2:04 AM

✗ #5

Aug 27, 2023, 2:01 AM

✗ #4

Aug 27, 2023, 1:34 AM

✗ #3

Aug 27, 2023, 1:34 AM

✗ #2

Aug 27, 2023, 1:34 AM

✗ #1

Aug 27, 2023, 1:29 AM

Atom feed for all

Atom feed for failures

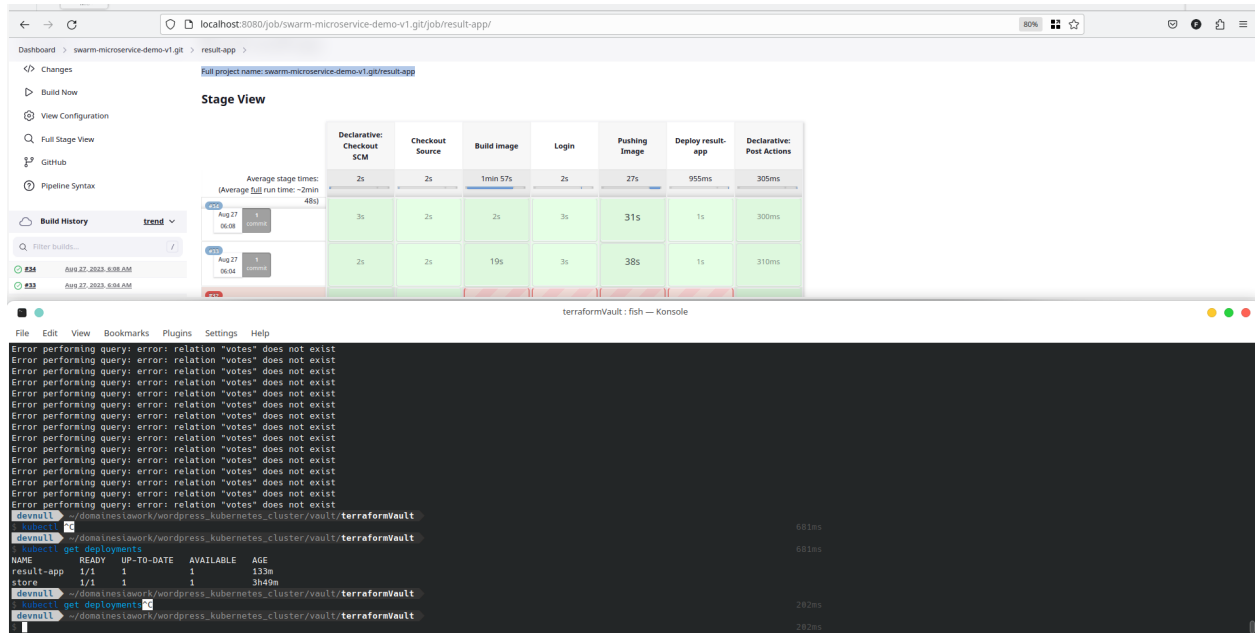
Branch postgresql

Full project name: swarm-microservice-demo-v1.git/postgresql

Stage View

		Declarative: Checkout SCM	Checkout Source	Deploying postgres-db
Average stage times: (Average full run time: ~10s)		2s	2s	1s
#9	Aug 27 02:19 1 commit	2s	2s	2s
#8	Aug 27 02:14 1 commit	2s	2s	1s failed
#7	Aug 27 02:12 No Changes	2s	2s	5s failed
#6	Aug 27 02:04 No Changes	2s	1s	415ms failed
#5	Aug 27 02:01 1 commit	2s	2s	570ms failed
#4	Aug 27 01:34 No Changes	4s	2s	137ms failed
#3	Aug 27 01:34 No Changes	2s	1s	95ms failed
#2	Aug 27 01:34 4 commits	2s	2s	125ms failed
#1				

4. Lalu part aplikasi yang saya deploy pertama adalah result-app. Saya membuat branch result-app



Hasil pipelinenya adalah seperti di atas. Untuk result-app ini ada beberapa masalah, yaitu error Unexpected strict mode reserved word. Sudah saya ubah nodejsnya, ubah versi socket.io, hapus node_modules melalui Dockerfile masih tidak berpengaruh. Tetapi saat ini hari sempat berjalan deployments-nya meskipun di logsnya tidak dapat terhubung ke redis. Tetapi karena file yang sudah berhasil dideploy tidak sengaja saya hapus saat ini hari maka saat dideploy error kembali meskipun dengan settingan yang sama.

```
kubectl logs result-app-69bf45c995-985bj

/node_modules/socket.io/node_modules/socket.io-client/node_modules/engine.io-client/node_modules/xmlhttprequest-ssl/lib/XMLHttpRequest.js:627
    for (let i = 0, len = listeners[event].length; i < len; i++) {
    ^^^
SyntaxError: Unexpected strict mode reserved word
    at Module._compile (module.js:439:25)
    at Object.Module._extensions..js (module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load (module.js:312:12)
    at Module.require (module.js:364:17)
    at require (module.js:380:17)
    at Object.<anonymous> (/node_modules/socket.io/node_modules/socket.io-client/node_modules/engine.io-client/lib/transports/index.js:5:22)
    at Module._compile (module.js:456:26)
    at Object.Module._extensions..js (module.js:474:10)
    at Module.load (module.js:356:32)
```

Di result app ini saya ubah Dockerfile terutama dengan menambahkan `rm -rf /node_modules` dan ubah port dari 80 menjadi 8080 agar dapat menggunakan security context dari kubernetes dan tidak berjalan sebagai root. Lalu saya buat deployment seperti berikut:

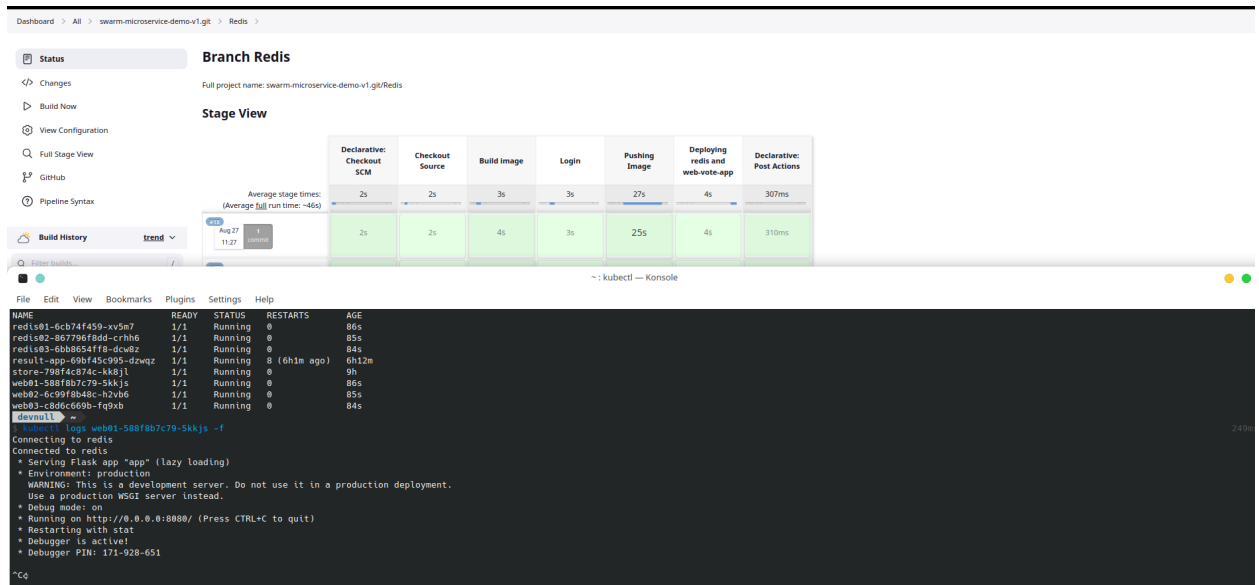
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: result-app
spec:
  selector:
    matchLabels:
      app: result-app
```

```
template:
  metadata:
    labels:
      app: result-app
  spec:
    securityContext:
      runAsUser: 1001
    containers:
      - name: result-app
        image: findnull45/result-app:latest
        securityContext:
          runAsUser: 1001
        allowPrivilegeEscalation: false
        resources:
          requests:
            memory: "64Mi"
            cpu: "150m"
          limits:
            memory: "1Gi"
            cpu: "1000m"
        ports:
          - containerPort: 8080
```

```
apiVersion: v1
kind: Service
metadata:
  name: result-app
spec:
  selector:
    app: result-app
  ports:
    - port: 8080
```

5. Selanjutnya yang saya lakukan adalah membuat deployment dari web-vote-app. Pada web-vote-app ini saya membuat tiga deployment yang isinya adalah deployment dari redis dan aplikasi python dari web-vote-app serta masing-masing service dari redis instance. Saya buat 3 deployment berbeda karena membutuhkan tiga suffix dua angka yang berbeda beda. Saya tidak menggunakan statefulset karena akan boros ketika sewaktu waktu ingin diexpose servicenya. Karena file yamlnya cukup panjang hanya saya sertakan linknya saja

<https://github.com/nizarakbarm/swarm-microservice-demo-v1/tree/master/web-vote-app>



Untuk hasil pipeline web-vote-app yang sudah berhasil dapat anda cek di atas.

- Selanjutnya saya membuat deployment dari vote-worker. Untuk vote-worker ini komposisinya hanya deployment tanpa service sebab vote-worker ini merupakan worker perantara yang tugasnya mengambil data dari postgresql dan cek data di redis. Untuk semua deployment dan dockerfile yang memungkinkan berjalan tanpa root di sini saya atur semuanya dengan user 1001 dan saya atur securityContext baik di pod maupun container. Di tiap deployment juga saya sertakan resource limit baik itu requests (soft limit) maupun limits (hard limit). Kembali ke vote-worker, pada vote-worker saya melakukan pengubahan base image yang sebelumnya *FROM java:7* menjadi *FROM maven:3.6.0-jdk-7-slim* sebab untuk *java:7* sudah tidak ada di *registry public docker*. Lalu saya set *USER 1001* juga agar tidak menggunakan root saat menjalankannya. Untuk file yml dari deploymentnya adalah sebagai berikut:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vote-worker
spec:
  selector:
    matchLabels:
      app: vote-worker
  template:
    metadata:
      labels:
        app: vote-worker
    node: worker0
```

```
spec:
  securityContext:
    runAsUser: 1001
  containers:
  - name: vote-worker
    image: findnull145/vote-worker:latest
  env:
  - name: FROM_REDIS_HOST
    value: "1"
  - name: TO_REDIS_HOST
    value: "3"
  - name: WORKER_NUMBER
    value: '01'
  securityContext:
    runAsUser: 1001
    allowPrivilegeEscalation: false
  resources:
    requests:
      memory: "64Mi"
      cpu: "50m"
    limits:
      memory: "1Gi"
      cpu: "2000m"
```

Dashboard > All > swarm-microservice-demo-v1.git > vote-worker

Status Branch vote-worker

Full project name: swarm-microservice-demo-v1.git/vote-worker

Stage View

	Declarative: Checkout SCM	Checkout Source	Build image	Login	Pushing Image	Deploy vote-worker	Declarative: Post Actions
Average stage times: (Average full run time: ~25s)	2s	2s	33s	1s	8s	507ms	301ms
Aug 27 13:31 No Changes	2s	2s	3s	3s	9s	1s	306ms

Build History trend

File Edit View Bookmarks Plugins Settings Help

```
devnull ~
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
redis01-6cb74f169-xv5e7             1/1     Running   0           3m58s
redis02-867736f8dd-crth6            1/1     Running   0           3m49s
redis03-6bb8654ff8-dcw8z            1/1     Running   0           3m48s
result-app-69bf45c995-dzwqz         1/1     Running   8 (6h3m ago) 6h15m
store-798f4c874c-kk8jl              1/1     Running   0           9h
vote-worker-5f9c7b65f6-7fvbz        1/1     Running   0           28s
web01-588f8b7c79-5kkjs              1/1     Running   0           3m58s
web02-6c99f8b48c-h2vb6              1/1     Running   0           3m49s
web03-c8dc669b-fq9xb               1/1     Running   0           3m48s
devnull ~
$ kubectl logs vote-worker-5f9c7b65f6-7fvbz -f
Can not write to /root/.m2/copy_reference_file.log. Wrong volume permissions? Carrying on ...
3 redis hosts
redisHosts[0] = 'redis01'
redisHosts[1] = 'redis02'
redisHosts[2] = 'redis03'
Connected to redis
Connected to redis
Connected to redis
Watching vote queue
```

Sedangkan untuk hasil pipeline dan cek logs dari podnya ada di atas. Jika dari logs pod-nya tersebut untuk podnya sudah berjalan dan pipelinenya juga berhasil.

Note: untuk masing-masing bagian tersebut sebelumnya pipeline dijalankan terpisah saat saya push pengerjaan di masing-masing branch. Karena dari saya belum memahami custom path *pipeline*. Metode *test*-nya adalah sementara saya pindahkan file dari projectnya ke *root directory* dari *github*. Lalu push ke branch-nya. Setelah itu dilakukan *Scan Repository Now* dilanjutkan dengan cek *Scan Repository Log*. Karena otomatis dilakukan *building* oleh *jenkins*, maka dilanjutkan dengan cek *Build History* dan cek *pipeline terbaru*.

7. Di sini juga saya *terraform* dari *MetalLB* pada https://github.com/nizarakbarm/swarm-microservice-demo-v1/tree/master/terraform_metalb yang niatnya ingin digunakan sebagai basis dari alokasi ip *ingress nginx*. Untuk *terraform*nya sudah berhasil di-*deploy*.

```
devnull ~ /swarm-microservice-demo-v1/terraform_metalb/environments/prod
$ terraform apply -target=module.preMetalLB

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

# module.preMetalLB.helm_release.metalb will be created
+ resource "helm_release" "metalb" {
+   atomic                = false
+   chart                 = "metalb"
+   cleanup_on_fail       = false
+   create_namespace      = false
+   dependency_update     = false
+   disable_crds_hooks    = false
+   disable_openapi_validation = false
+   disable_webhooks      = false
+   force_update          = false
+   id                    = (known after apply)
+   lint                  = false
+   manifest               = (known after apply)
+   max_history            = 0
+   metadata               = (known after apply)
+   name                  = "metalb"
+   namespace              = "metalb-system"
+   pass_credentials      = false
+   recreate_pods         = false
+   render_subchart_notes = true
+   replace               = false
+   repository             = "https://metalb.github.io/metalb"
+   reset_values           = false
+   reuse_values           = false
+   skip_crds              = false
+   status                 = "deployed"
+   timeout                = 300
+   verify                 = false
+   version                = "0.13.10"
+   wait                   = true
+   wait_for_jobs          = false
+ }

# module.preMetalLB.kubernetes_namespace.metalb_namespace will be created
+ resource "kubernetes_namespace" "metalb_namespace" {
+   id = (known after apply)

+   metadata {
+     generation = (known after apply)
+     labels     = {
+       "pod-security.kubernetes.io/audit" = "privileged"
+       "pod-security.kubernetes.io/enforce" = "privileged"
+       "pod-security.kubernetes.io/warn"   = "privileged"
+     }
+     name = "metalb-system"
+   }
+   resource_version = (known after apply)
+ }
```



```

+ labels = {
+   "pod-security.kubernetes.io/audit" = "privileged"
+   "pod-security.kubernetes.io/enforce" = "privileged"
+   "pod-security.kubernetes.io/warn" = "privileged"
+ }
+ name = "metallb-system"
+ resource_version = (known after apply)
+ uid = (known after apply)
}
}

Plan: 2 to add, 0 to change, 0 to destroy.

Warning: Resource targeting is in effect

You are creating a plan with the -target option, which means that the result of this plan may not represent all of the changes requested by the current configuration.

The -target option is not for routine use, and is provided only for exceptional situations such as recovering from errors or mistakes, or when Terraform specifically suggests to use it as part of an error message.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

module.preMetalLB.kubernetes_namespace.metallb_namespace: Creating...
module.preMetalLB.kubernetes_namespace.metallb_namespace: Creation complete after 0s [id=metallb-system]
module.preMetalLB.helm_release.metallb: Creating...
module.preMetalLB.helm_release.metallb: Still creating... [10s elapsed]
module.preMetalLB.helm_release.metallb: Still creating... [20s elapsed]
module.preMetalLB.helm_release.metallb: Still creating... [30s elapsed]
module.preMetalLB.helm_release.metallb: Still creating... [40s elapsed]
module.preMetalLB.helm_release.metallb: Still creating... [50s elapsed]
module.preMetalLB.helm_release.metallb: Still creating... [1m0s elapsed]
module.preMetalLB.helm_release.metallb: Still creating... [1m10s elapsed]
module.preMetalLB.helm_release.metallb: Creation complete after 1m14s [id=metallb]

Warning: Applied changes may be incomplete

The plan was created with the -target option in effect, so some changes requested in the configuration may have been ignored and the output values may not be fully updated. Run the following command to verify that no other changes are pending:

  terraform plan

Note that the -target option is not suitable for routine use, and is provided only for exceptional situations such as recovering from errors or mistakes, or when Terraform specifically suggests to use it as part of an error message.

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

```

Tetapi karena di rke2-server masih belum menemukan cara mengatasi konflik dari ingress bawaan rke2-server saya batalkan untuk implementasinya.

Untuk masing-masing github repo dari jawaban soalnya adalah

<https://github.com/nizarakbarm/bashPrimeNumber>

<https://github.com/nizarakbarm/swarm-microservice-demo-v1>