Úvod

Účelom je splniť nasledujúce zadanie, ktoré simuluje prácu u zákazníka. V dokumente sú špecifikované základné informácie. Pokiaľ potrebujete doplňujúce informácie, chovajte sa tak akoby ste pracovali u zákazníka a danú informáciu si vyžiadajte (prostredníctvom email-u/microsoft teams-u).
Hlavným účelom zadania je overiť si samostatnosť a spôsob riešenia problému.

Kontakt

1. **Microsoft Teams**
2. **Email:**
    - daniel.rajcan@unicorn.com
    - michal.okresa@unicorn.com
    - marek.moravcik@fri.uniza.sk - v prípade nefunkčného OpenStack prostredia

Zadanie

Zákazník chce vytvoriť Kubernetes cluster na privátnom/komunitnom cloud-e (Openstack) Katedry Informačných Sietí v Žiline. Hlavnou zákazníckou požiadavkou je, aby všetko bolo pripravené v duchu DevOps, pričom kladie veľký dôraz na automatizáciu.

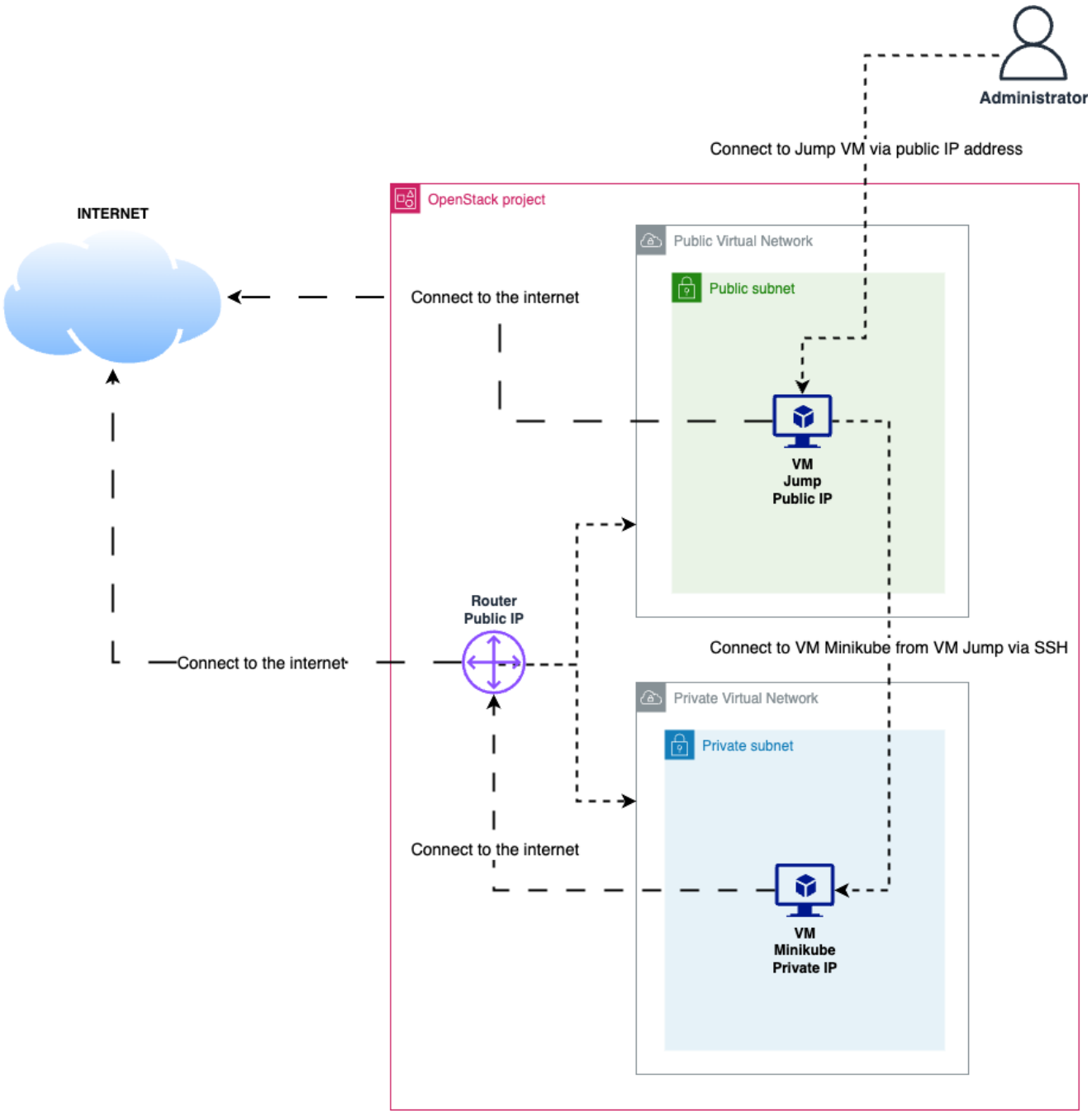
Ďalej zákazník kladie dôraz na:

- Všetko je realizované v prostredí Openstack: https://158.193.152.44/horizon
- Hlavný IaC nástroj je Terraform: https://www.terraform.io/
- Všetky zdrojové kódy sú uložené a verzionované v GIT repozitáry: https://github.com/
- Docker images sú uložené v Docker Hub registry: https://hub.docker.com/
- Systematická menná konvencia pre aplikácie a kubernetes resources
- Bezpečnosť senzitívnych dát (aj mená, heslá a tokeny sa považujú za citlivé dáta)
- Vysoká dostupnosť
- Úplne sa vyhnúť manuálnej konfigurácii.
- Dokumentácia vypracovaného zadania (stačí použiť README file v GIT repository)

1 Infrastructure

Cieľom prvej časti finálneho zadania je vytvoriť infraštruktúru pomocou nástroja Terraform v prostredí OpenStack v ktorom bude prevádzkovaná kontajnerová aplikácia na platforme Kubernetes.

Na nasledujúcom diagrame je zobrazená high-level architektúra zadania.



GIT

Pre účely vytvorenia požadovanej architektúry si vytvorte vlastné GitHub repozitáre, ktorých obsah bude hodnotený po odovzdaní kompletného zadania.

Nasledujúci repozitár môžete použiť ako inšpiráciu pre vytvorenie Vašich Terraform skriptov:

- https://github.com/daniel-rajcan/kis-onpk - tento repozitár považujte za pomôcku, neodporúčam odovzdávať zadanie, ktoré bude len kópia už vytvorených skriptov.

Terraform

- Vytvorený kód musí obsahovať dva moduly pre sieťovú (network) a výpočtovú vrstvu (compute),
- Kód musí obsahovať parametre tak aby ho bolo možné spustiť pre viaceré prostredia a nedochádzalo ku konfliktom jednotlivých konfigurácií,
- Pre inštaláciu Kubernetes cluster-a použite nástroj minikube https://minikube.sigs.k8s.io/docs/start/, ktorý je prevádzkovaný na jednej virtuálnej inštancii. Použite poslednú dostupnú verziu s tým, že verziu bude možné v čase meniť pomocou parametrov v Terraforme,
- Minikube kubernetes cluster nastavte tak aby bol spustený jeden master a dva worker nodes na jednej virtuálnej inštancii,
- Do konfigurácie pridajte príkaz "**minikube addons enable ingress**", ktorý zabezpečí aktiváciu plugin-u s ktorým budete pracovať neskôr,
- Inštalácia a konfigurácia Kubernetes cluster-a musí byť plne automatizovaná,
- Security group pre Jump musí byť konfigurovateľný tak aby povoľoval komunikáciu z vybraných IP adries.
- Pre sprístupnenie aplikácie, ktorá bude spustená na Minikube v neskorších fázach nastavte iptables na Jump servery tak aby cez SNAT a DNAT sprístupnili aplikáciu prevádzkovanú na Minikube servery. Druhou možnosťou je spraviť SSH tunel z Vášho lokálneho počítača,
- Konfigurácia Jump servera musí byť automatizovaná,
- **Bonusové body:** Terraform nečaká na to kým sa konfigurácia virtuálnej inštancie dokončí. To znamená, že príkaz terraform apply skončí bez toho aby počkal na to, či sa ukončí konfigurácia virtuálnej inštancie správne. Skúste navrhnúť a implementovať riešenie, ktoré zabezpečí, že príkaz terraform apply skončí až vtedy, keď je virtuálna inštancia správne nakonfigurovaná.

Docker Hub repository

Zaregistrujte sa na https://hub.docker.com a vytvorte si vlastné docker hub repozitáre ako úložisko pre vytvorené docker images.

Docker (Container images)

Naklonujte si git repozitáre:

- https://github.com/michaello1/zauni-zadanie-appbackend
- https://github.com/michaello1/zauni-zadanie-appfrontend

Vašou úlohou bude doplniť predpripravené Dockerfiles, aby bolo možné vytvoriť container images, ktoré následne uložíte do vašich docker hub repozitárov, ktoré ste si vytvorili. Inštrukcie, čo je potrebné doplniť máte uvedené vo forme komentárov aj priamo v Dockerfiles (`zauni-zadanie-appbackend/Dockerfile`, `zauni-zadanie-appfrontend/Dockerfile`)

Appbackend

Vašou úlohou je:

1. pridať príkaz, ktorý skopíruje zo stage **builder** súbor: /builder/main do adresára /app/ (všimnite si na riadku č.8, že výstupom kompilácie je súbor s názvom main).
2. nastaviť pracovný adresár (workdir) na: /app
3. pridať inštrukciu, ktorá zabezpečí, že po vytvorení kontajnera sa spustí aplikácia (tj. ./main).

Appfrontend

Vašou úlohou je:

4. pridať inštrukciu, ktorá bude informovať o tom, že kontajner publikuje TCP port 8080.
5. pridať inštrukciu, ktorá zabezpečí, že po vytvorení kontajnera sa spustí nginx:

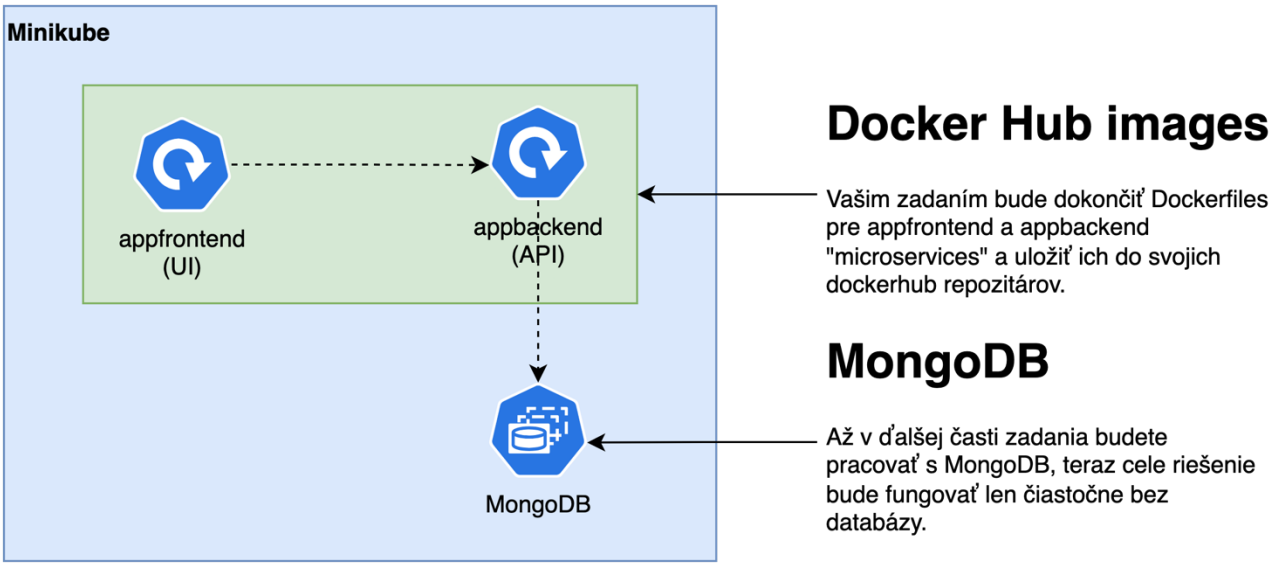
   ```
   nginx -g daemon off;
   ```

   TIP: pozor ako zadávate parametre do inštrukcie

Bonusové body: vytvorte tagy pre docker images pomocou sémantického verzovanie (https://semver.org/).

Pozor, nezabudnite, že zmeny v Dockerfile nemôžete commit-núť do GitHub repozitárov, ktoré su tu uvedené, ale musíte mať vytvorené vlastné repozitáre, ktoré budú slúžiť na odovzdanie celého zadania.

„Microservices“ appbackend a appfrontend budeme používať aj v ďalších častiach zadania a spolu s MongoDB databázou budú tvoriť jednoduchú aplikáciu, ktorou si overíte koncepty, ktoré si prechádzame na prednáškach.



Vzhľadom na to, že v tomto štádiu ešte neviete ako si jednotlivé časti nasadiť do Kubernetes clustra a nemáte ani MongoDB, tak si môžete overiť, že vaše container images sú správne týmto spôsobom:

- appbackend – spustite si kontajner z image, ktorý ste vytvorili (docker run ...). Správny výstup je (aplikácia skončí s chybou, pretože sa nevie pripojiť k databáze):

```
version 1.0.1
serving on port 9080...
tests:
curl -s localhost:9080/ok
curl -s localhost:9080/platforms
curl -s localhost:9080/platforms | jq .
MongoDB connection details:
MONGO_CONN_STR:mongodb://localhost:27017/?replicaSet=rs0&connect=direct
MONGO_USERNAME:admin
MONGO_PASSWORD:
attempting mongodb backend connection...
2023/11/06 15:08:37 couldn't connect to the databaseserver selection error: server selection timeout
current topology: Type: Single
Servers:
Addr: localhost:27017, Type: Unknown, State: Connected, Average RTT: 0, Last error: connection(localhost:27017[-121]) connection is closed
```

- appfrontend – spustite si kontajner z image, ktorý ste vytvorili (docker run ...), pridajte aj parameter -p8080:8080. Správny výstup je:

```
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/env.sh
window._env_ = {
  REACT_APP_APIHOSTPORT: ""
}
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/11/06 15:11:50 [notice] 1#1: using the "epoll" event method
2023/11/06 15:11:50 [notice] 1#1: nginx/1.24.0
2023/11/06 15:11:50 [notice] 1#1: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git20220924-r4)
2023/11/06 15:11:50 [notice] 1#1: OS: Linux 5.15.0-88-generic
2023/11/06 15:11:50 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/11/06 15:11:50 [notice] 1#1: start worker processes
2023/11/06 15:11:50 [notice] 1#1: start worker process 26
2023/11/06 15:11:50 [notice] 1#1: start worker process 27
2023/11/06 15:11:50 [notice] 1#1: start worker process 28
2023/11/06 15:11:50 [notice] 1#1: start worker process 29
```

- čiastočnú funkčnosť aplikácie si môžete overiť otvorením URL: http://localhost:8080 v prehliadači. Aplikácia sa nebude vedieť pripojiť k appbackend API, takže sa zobrazí len hlavička.

Helm charts

Úlohou tejto časti je vytvoriť helm chart ktorým bude možné nasadiť frontend a backend časť aplikácie pre ktoré ste vytvorili docker image. Helm chart musí spĺňať základné štandardy ako popis vstupných premenných ako súčasť readme a rovnako vhodne definovaný výstup NOTES.txt. Výstup uložte do zložky charts vášho repozitára.

App frontend:

- Vytvorenie deploymentu pre ktorý bude možné zadefinovať počet inštancií a konfiguráciu samotnej FE aplikácie vhodným spôsobom.
    - Hint: pre konfiguráciu môžete použiť premennú env typu pole ktorú použijete v šablóne.
- Vytvorenie service aby bolo možné pristúpiť na FE aplikáciu z prehliadača pracovnej stanice.
- Aplikácia očakáva environment premennú: **REACT_APP_APIHOSTPORT**

App Backend

- Vytvorenie deploymentu pre ktorý bude možné zadefinovať počet inštancií a konfiguráciu
- Vytvorenie service aby bolo možné pristúpiť na BE aplikáciu z prehliadača pracovnej stanice.
- Použiť niektorý z existujúcich helm chartov pre Mongo DB a zakomponovať ako subchart, ktorý deployne mongo db pre potreby backend aplikácie.
    - Hint: Použite mongodb z https://charts.bitnami.com/bitnami
- Environment premenné, ktoré potrebujete správne nastaviť: **MONGO_CONN_STR** (napr.: mongodb://appbackend-mongodb:27017/dynamicky_nazov_db), **MONGO_USERNAME**, **MONGO_PASSWORD**

Bonusové body: zabezpečte, že všetky konfiguračné parameter budú validované, že sú správne zadané aby sme mali istotu, že nasadená aplikácia bude nakonfigurovaná správne. Pre obe časti aplikácie (FE a BE) použite jeden univerzálny helm chart.

CI/CD

Úlohou tejto časti je vytvorenie dvoch CI a dvoch CD Tekton pipelines v ktorých využijete Dockerfiles a Helm charts z predchádzajúcich častí práce. Pri vytváraní Tekton objektov nezabudnite na dodržiavanie Kubernetes štandardov. Výstup uložte do adresára *cicd* vášho repositára.

CI Pipelines:

- Build frontend časti aplikácie a push do vlastného Docker Hub repository
    - použite *git-clone* a *buildah* (alebo *kaniko*) tasky z Tekton katalógu
    - nezabudnite na git a docker credentials v secrets
- Build backend časti aplikácie a push do vlastného Docker Hub repository
    - použite *git-clone* a *buildah* (alebo *kaniko*) tasky z Tekton katalógu
    - nezabudnite na git a docker credentials v secrets

CD Pipelines:

- Nasadenie frontend časti aplikácie prostredníctvom Helm chart
    - Použite *git-clone* a *helm-upgrade-from-source* tasky z Tekton katalógu
- Nasadenie backend časti aplikácie prostredníctvom Helm chart
    - Použite *git-clone* a *helm-upgrade-from-source* tasky z Tekton katalógu

Bonusové body: pipeliny sa budú spúšťať na základe push eventu do git repositára, live prezentácie riešenia