**terraform 團隊協作的問題**

* terraform 是 local apply 的 command tool
  + 每個人都在自己的 local 電腦上 apply terraform
    - 同時改一樣的東西就會 conflicts
  + Iac 的原則是 infrastructureas code，然而各自 apply 卻違反這樣的原則
    - 理想上 master branch 就是最新的 infrastructure，然而因為需要手動 apply，讓 infra 可能停留在前幾個 commit
    - 或是 feature branch 已經 apply 到 infra 上
* IaC 後，需要自動化的 PR Review
  + 最好自動化產生 plan 結果當作 Review 依據
* 最好有自動化的 auto terraform apply
  + 由 CI / CD trigger，依循 git-flow 流程，plan 與 apply infrastructure
  + 透過工具 apply，避免人為 apply 時造成的 human error

**Why not Github Action**

如果使用 github.com，完全可以使用 Github Action，來執行 terraform automation，除了本課程內有提供範例外，網路上已經有更多的範例可以參考

然而有些 version control system 是不方便使用 Github Action

* 使用 self-host github enterprise，開啟 Github Action 要耗費大量的算力，目前仍有效能問題
* 使用其他 version control system 例如 gitlab，bitbucket...或是上述 vcs 的 self-hosted enterprise 版本，自然就沒有 Github Action
* 或是團隊不想用 Github Action

這時可以考慮使用公司的 CI / CD 系統，自行寫 terraform 的 pipeline，然後整理前述課程使用到的 command 寫成 shell script 執行。只要是知名的 CI / CD 工具，都能找到許多 terraform 的 pipeline 範例，而這些範例多半是共通的。

如果不希望自己維護 terraform pipeline，現在已經有開源版本的整合工具，幫你做自動化，就是這張要介紹的 [Atlantis: Pull Request Automation](https://www.runatlantis.io/)

參考 terraform-30-days 上 [實際執行的 PR 範例](https://github.com/chechiachang/terraform-30-days/pull/6)

**About atlantis**

[Atlantis](https://www.runatlantis.io/) 是一款開源免費的自動化 terraform 工具。基本工作流程很單純

* PR 產生的時候， atlantis 自動執行 terraform plan
* Merge 到 master 之後，依據設定，atlantis 執行 terraform apply

Features

* Self-hosted
* 已支援許多 CVS，包含 Github, bitbucket, gitlab, azure-devops, ...

**Local Run atlantis**

官方說明文件試跑 [Atlantis Local Run](https://www.runatlantis.io/guide/testing-locally.html#testing-locally)，主要步驟為

* 安裝 terraform
* 使用 ngrok 暫時產生一個 dns forwarding 到本地的 4141 port
* 設定 github webhook 將 event 從 github 推到本地的 atlantis
* 設定 github personal access token，讓本地的 atlantis server 可以將 plan 結果送到 github comment
* 本地啟動 atlantis server，使用上述參數
* 到 Github 發 PR，使用 comment 控制 atlantis

Install atlantis

wget https://github.com/runatlantis/atlantis/releases/download/v0.17.2/atlantis\_darwin\_amd64.zip

unzip atlantis\_darwin\_amd64.zip

sudo mv atlantis /usr/local/bin/atlantis

atlantis -h

Install ngrok and run

* 記下 forwarding url ，作為 URL 環境變數
* 保持 ngrok 執行的狀況下，開新 terminal 執行下列步驟

wget https:*//bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-darwin-amd64.zip*

unzip ngrok-stable-darwin-amd64.zip

sudo mv ngrok /usr/local/bin/ngrok

./ngrok http 4141

...

Forwarding http:*//41eb-123-194-159-122.ngrok.io -> http://localhost:4141*

...

Config Github Webhook

* githob repository (ex. chechiachang/terraform-30-days)
* settings -> webhook -> add webhook
* application/json
* <https://41eb-123-194-159-122.ngrok.io/events>
* 記下 webhook token，放在安全的地方 -> 作為 SECRET 環境變數

Create Github Personal Access token

* github user -> settings -> Developer settings -> Personal access tokens
* atlantis
* 只是本地試用，expiration 選 7 天
* 記下 access token，放在安全的地方 -> 作為 TOKEN 環境變數

Local Run atlantis server

* 保持 atlantis server 執行

export URL="https://...................ngrok.io"

export SECRET="ep.................quh"

export TOKEN="ghp\_..........................."

export USERNAME="chechiachang"

export REPO\_ALLOWLIST="github.com/chechiachang/terraform-30-days"

atlantis server \

--atlantis-url="${URL}" \

--gh-user="${USERNAME}" \

--gh-token="${TOKEN}" \

--gh-webhook-secret="${SECRET}" \

--repo-allowlist="$REPO\_ALLOWLIST"

...

{"level":"info","ts":"2021-08-31T23:03:31.616+0800","caller":"server/server.go:680","msg":"Atlantis started - listening on port 4141","json":{}}

**Use atlantis**

[Atlantis Official Doc: Usage](https://www.runatlantis.io/docs/using-atlantis.html#atlantis-help)

* help
* plan
* apply

在 [Github terraform-30-days](https://github.com/chechiachang/terraform-30-days/pull/6" \t "_blank) 的範例

**atlnatis help**

comment atlantis help

atlantis

Terraform Pull Request Automation

...

**atlnatis plan**

comment atlantis plan -d azure/\_poc/compute/

* 這邊使用 -d 指定想要 plan 的 root module 的 directroy 進行 plan
* 這邊使用的是 local 本機的 credential，如果有 az login 的 credential 就可以 apply，沒有就會失敗
  + 如果使用 remote vm / k8s 跑 atlantis server，就需要遠端設定 credential
* 從 server log 可以看到
  + github webhook 的 json event
  + event parse 看 comment 內容有無 help, plan, 或 apply command 及參數
  + 如有就執行 help, plan 或 apply
  + 使用 Github personal access token 再打回去 Github API，產生 comment

# atlantis log

# parse comment as command

{"level":"info","ts":"2021-08-31T23:47:23.483+0800","caller":"events/events\_controller.go:417","msg":"parsed comment as command=\"plan\" verbose=false dir=\"azure/\_poc/compute\" workspace=\"\" project=\"\" flags=\"\"","json":{}}

# acquired lock with id

{"level":"info","ts":"2021-08-31T23:47:24.378+0800","caller":"events/project\_locker.go:80","msg":"acquired lock with id \"chechiachang/terraform-30-days/azure/\_poc/compute/default\"","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

# terraform init

{"level":"info","ts":"2021-08-31T23:47:31.435+0800","caller":"terraform/terraform\_client.go:280","msg":"successfully ran \"/Users/che-chia/.asdf/shims/terraform init -input=false -no-color\" in \"/Users/che-chia/.atlantis/repos/chechiachang/terraform-30-days/6/default/azure/\_poc/compute\"","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

# terraform workspace

{"level":"info","ts":"2021-08-31T23:47:31.959+0800","caller":"terraform/terraform\_client.go:280","msg":"successfully ran \"/Users/che-chia/.asdf/shims/terraform workspace show\" in \"/Users/che-chia/.atlantis/repos/chechiachang/terraform-30-days/6/default/azure/\_poc/compute\"","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

# terraform plan

{"level":"info","ts":"2021-08-31T23:47:48.746+0800","caller":"terraform/terraform\_client.go:280","msg":"successfully ran \"/Users/che-chia/.asdf/shims/terraform plan -input=false -refresh -no-color -out \\\"/Users/che-chia/.atlantis/repos/chechiachang/terraform-30-days/6/default/azure/\_poc/compute/default.tfplan\\\"\" in \"/Users/che-chia/.atlantis/repos/chechiachang/terraform-30-days/6/default/azure/\_poc/compute\"","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

# policy check

{"level":"info","ts":"2021-08-31T23:47:50.017+0800","caller":"events/plan\_command\_runner.go:214","msg":"Running policy check for command=\"plan\" verbose=false dir=\"azure/\_poc/compute\" workspace=\"\" project=\"\" flags=\"\"","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

{"level":"info","ts":"2021-08-31T23:47:50.017+0800","caller":"events/policy\_check\_command\_runner.go:36","msg":"no projects to run policy\_check in","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

* [Github plan 結果在這邊](https://github.com/chechiachang/terraform-30-days/pull/6#issuecomment-909345733)
* 其實跟本地 terraform plan 一樣

Terraform used the selected providers to generate the following execution

plan. Resource actions are indicated with the following symbols:

+ create

<= read (data resources)

Terraform will perform the following actions:

*# module.linuxservers.data.azurerm\_public\_ip.vm[0] will be read during apply*

*# (config refers to values not yet known)*

<= data "azurerm\_public\_ip" "vm" {

...

}

Plan: 9 to add, 0 to change, 0 to destroy.

**atlantis apply**

Apply 也很單純，就是 apply

* comment: atlantis apply -d azure/\_poc/compute/

注意：這邊 apply 下去就會自動 apply，沒有 double comfirm yes or no 了

{"level":"info","ts":"2021-08-31T23:38:43.963+0800","caller":"events/events\_controller.go:417","msg":"parsed comment as command=\"apply\" verbose=false dir=\"azure/\_poc/compute\" workspace=\"\" project=\"\" flags=\"\"","json":{}}

{"level":"info","ts":"2021-08-31T23:38:45.151+0800","caller":"events/apply\_command\_runner.go:110","msg":"pull request mergeable status: true","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

{"level":"info","ts":"2021-08-31T23:38:45.157+0800","caller":"runtime/apply\_step\_runner.go:38","msg":"starting apply","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

{"level":"info","ts":"2021-08-31T23:40:05.868+0800","caller":"terraform/terraform\_client.go:280","msg":"successfully ran \"/Users/che-chia/.asdf/shims/terraform apply -input=false -no-color \\\"/Users/che-chia/.atlantis/repos/chechiachang/terraform-30-days/6/default/azure/\_poc/compute/default.tfplan\\\"\" in \"/Users/che-chia/.atlantis/repos/chechiachang/terraform-30-days/6/default/azure/\_poc/compute\"","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

{"level":"info","ts":"2021-08-31T23:40:05.868+0800","caller":"runtime/apply\_step\_runner.go:57","msg":"apply successful, deleting planfile","json":{"repo":"chechiachang/terraform-30-days","pull":"6"}}

**Workflow**

實際運作的 gitflow，大約是這樣

* push PR
* PR review
  + 執行 atlantis plan
* merge 進入 main
* automatically atlantis apply main，確保 remote infra 緊跟 main branch 的變更

**Pros & cons**

使用 atlnatis 有底下優缺點

* 不再需要工程再在本地 apply
* 安全性來說，工程師 local 電腦也不再需要 azure 權限的 credential。畢竟 terraform 所需的帳號權限還是擁有 plublic cloud 上許多資源的生殺大權。減少暴露到只有 atlantis server 上有
* 使用 github comment 控制

缺點是要多養一台或多台 atlantis server，然而 atlantis server 基本上是 stateless server，如果有 k8s 的話非常好養

**TODOs for production**

上面只是在本地電腦測試一下 atlantis 的功能，實際上如果要讓 production 環境使用，還有以下代辦事項要處理

* Deployment: 使用適合 production 環境的 VM / k8s 來執行 atlantis
  + k8s 上可以使用 [helm chart](https://github.com/runatlantis/helm-charts)
* High Availability: 執行多個 atlantis server replicas，當其中一個故障時整體 gitflow 功能仍有效
  + [helm chart 便可以設定](https://github.com/runatlantis/helm-charts/blob/main/charts/atlantis/values.yaml#L204)
  + 由於 terraform remote backend 有提供 state lock [複習第 2 章](https://ithelp.ithome.com.tw/articles/lecture/02-basic-state.md)，不會有多個 atlantis server apply 同一個 root module 的問題
* github personal access token 可以使用專屬的 bot user，而不要用 chechiachang
* credential 改用 azure service principal 的設定
  + 放在 atlantis server 可及的安全之處
  + 如果是 k8s 可以搭配 hashicorp vault 使用

**Config Atlantis**

atlantis default 使用 terraform cmd，然而本課程有許多範例使用 terragrunt，atlantis 也支援，需要底下額外設定

[Atlantis: terrragrunt support](https://www.runatlantis.io/docs/custom-workflows.html#use-cases)

（大家先自己研究，我有時間會來補的（汗））

**Homework**

* 依照範例，在本地電腦起一個 atlantis server
* 透過 github comment 操作，terragrunt plan 及 apply 任一 module

**Alternative: terraform cloud**

Terraform Cloud 是 terraform 官方提供的 Terraform automation Saas 服務

需要收費，[請見 terraform cloud pricing](https://www.hashicorp.com/products/terraform/pricing)。然而也提供更多強大的功能，除了 atlantis 的 remote plan 與 remote apply 外，還有私有 module registry，state file 版本控管...等功能

目前不是本課程推薦的解決方案，但 terraform cloud 不斷推陳出新許多新功能，未來值得期待。本課程會依據後續參賽進度調整，有機會再分享 terraform cloud 內容。