

# † bungsaufgaben

## † bung 1

In der ersten Aufgabe geht es darum das einen Jenkins Job fŕ die Ansible Playbooks einzurichten.

Jeder Teilnehmer erstellt seinen eigenen CI Job auf einer Jenkins Instanz (Die URL wird im Workshop bekannt gegeben). Als Erstes forkst du bitte das GitHub Repository <https://github.com/sparsick/ci-iaq-workshop>, so dass du deine eigene Repository erhŠlst.

In diesem Branch passt du den VM Namen in etwas deiner Wahl in den folgende Dateien an:

¥ `ci-test-vm/main.tf`: Zeile 14, der Wert `ansible-test-instance` muss geŠndert werden

¥ `install-hero-app.yml`: Zeile 1, der Wert `ansible-test-instance` muss geŠndert werden

Wichtig ist dabei, das der Name der VM in beiden Dateien gleich ist. ZusŠtzlich wird der Name des SSH Keys nach deiner Wahl angepasst:

¥ `ci-test-vm/main.tf`: Zeile 8

¥ `ci-test-vm/main.tf`: Zeile 19

Commite und pushe deine !nderung in dein Remote Git Repository.

```
git commit -m "change vm name"
git push --set-upstream origin main
```

Jetzt ist alles bereit, um den Jenkins Job fŕ dieses Ansible Playbook zu erstellen. Dazu melde dich im Jenkins (URL wird im Workshop bekannt gegeben) mit den Credentials:

¥ Username: `admin`

¥ Passwort: `admin123`

Du richtest eine neuen Pipeline Jenkins Job ein, in dem du auf `Element anlegen` klickst und `Pipeline` auswŠhlt.

Unter `Pipeline ! Definition` wŠhlt du `Pipeline script from SCM ! Git` aus und trŠgst deine Git Repo URL und den Branchnamen `exerise-01` an.

Damit wŠren die Grundlagen fŕ den Jenkins Job konfiguriert. Jetzt musst du die Pipeline in der `Jenkinsfile` Datei, die im Git Repository liegt, zu Ende schreiben.

Die Pipeline soll mit Terraform eine Test VM hochfahren, dann einmal das Ansible Playbook `install-hero-app.yml` gegen diese Test VM laufen lassen und am Ende, egal wie das Ergebnis war, die Test VM mit Hilfe von Terraform wieder lŠschen.

Hinweise:

- ¥ Terraform erstellt VMs mit `terraform apply -auto-approve -var="hcloud_token=${HPCLOUD_TOKEN}"`
- ¥ Terraform löscht VMs mit `terraform destroy -auto-approve -var="hcloud_token=${HPCLOUD_TOKEN}"`
- ¥ Ansible Playbook Run Kommando: `ansible-playbook -i inventory/test.hcloud.yml install-hero-app.yml`

## †bung 2

In dieser †bung sollst du mit Hilfe von Testinfra Tests für dein Ansible Playbook schreiben. Dafür wurde im Branch `exercise-02` unter `test/test_heroapp.py` ein Template für dich vorbereitet.

Nachdem du die Tests geschrieben hast, muss das Jenkinsfile so anpassen, dass deine Test VM richtig angesprochen wird.

Hinweise:

- ¥ !ndere in deinem Jenkins-Job den Branch auf `exercise-02`
- ¥ [TestInfra Beispiel](#)
- ¥ [Dokumentation der Testinfra Module](#)
- ¥ Testinfra Module, die nützlich sein können:
  - " `package`
  - " `service`
  - " `docker`
  - " `socket`
- ¥ Im Jenkinsfile muss der Hostname `ansible-test-instance` angepasst werden, so dass er dem `hosts` Eintrag in der Datei `install-hero-app.yml` entspricht.

## †bung 3

In dieser †bung baust du den Linter Ansible-Lint in deine CI-Pipeline ein.

Ausgangspunkt für diese †bung ist der Branch `exercise-03`. Der Linter soll die Ansible Playbook vor allen anderen Schritten überprüfen.

Hinweise:

- ¥ !ndere in deinem Jenkins-Job den Branch auf `exercise-03`
- ¥ Ansible Lint wird mit `ansible-lint my-playbook.yml` aufgerufen.
- ¥ [Ansible-Lint Homepage](#)