2.0 Installation

Contents of the current section:

Overview

In this section we will cover up installation process from scratch. Stack deployment will consists of two parts infrastructure and application

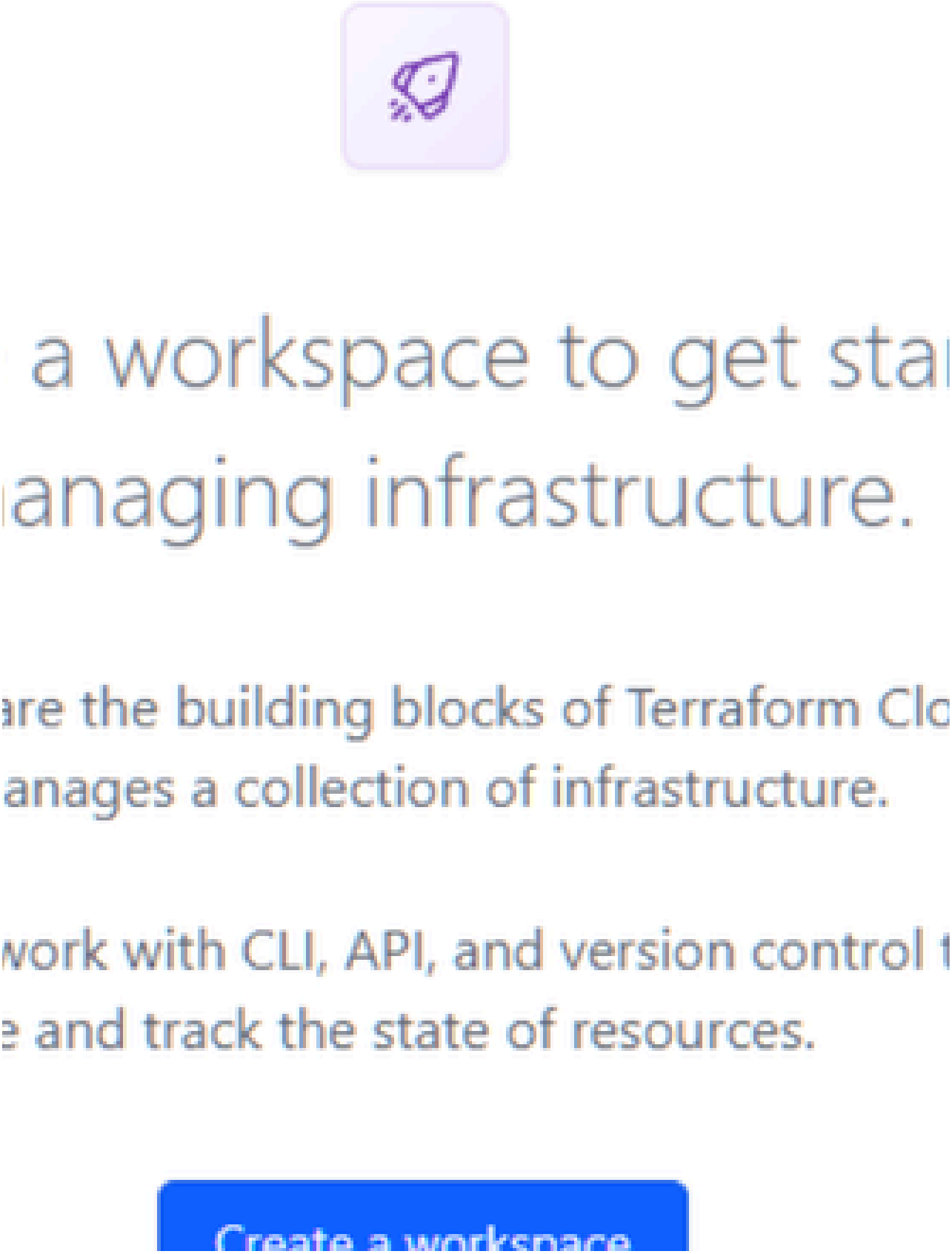
Requirements and tools

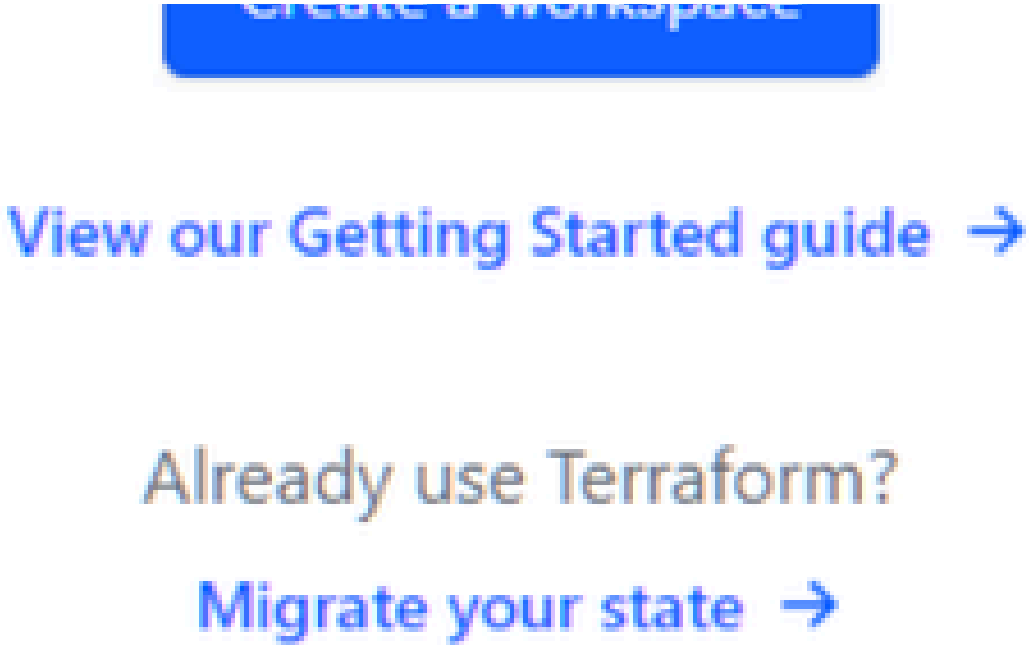
1. terraform cli
2. AWS account (where everything will be deployed)
3. AWS account where our host_zone is located
4. domain (example.com)
5. Cloud terraform account (app.terraform.io)
6. Repo access (GitHub - tntk-io/tntk-infra)

Infrastructure deployment

To deploy our terraform code first of all we need to create account on app.terraform.io and create organization and workspace in my example i created **opex-apex** organization

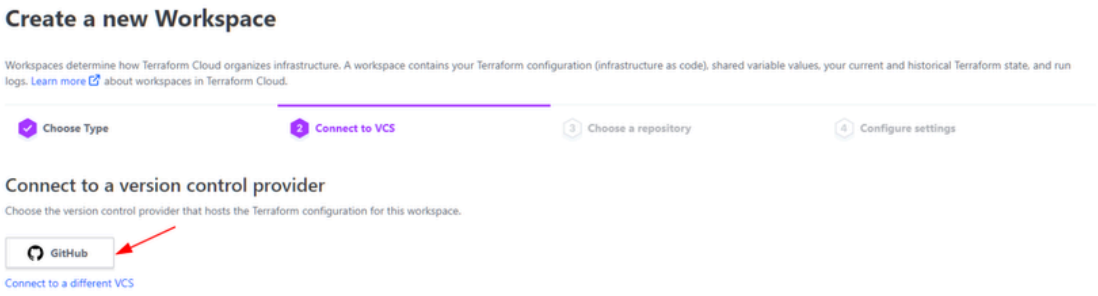
1. Login to the terraform cloud.



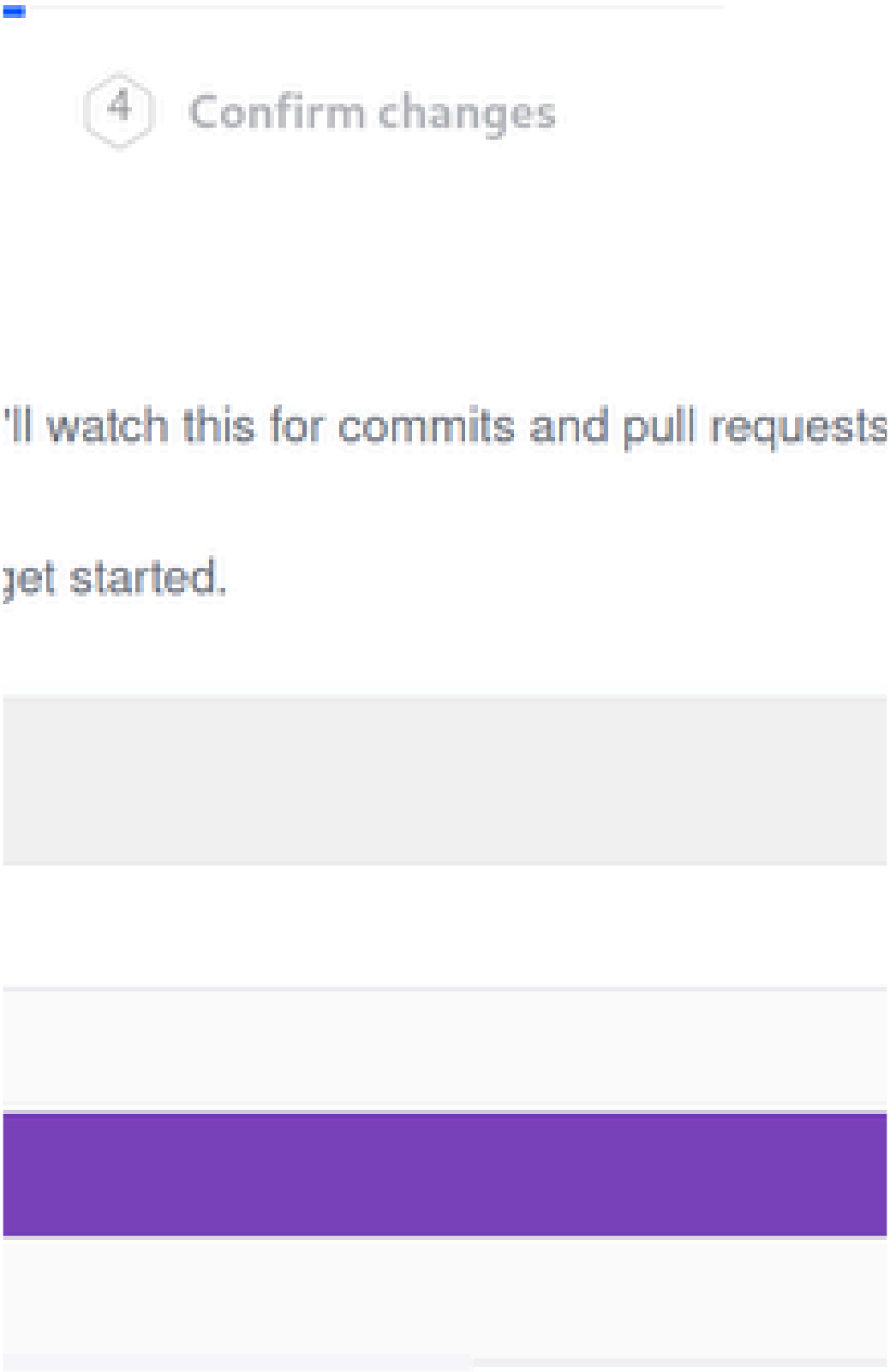


after that you need to create workspace. We need to go on **workspaces** → **new workspace**→ **create workspace** click on then Version control workflow and we have to connect it with our github repository where our terraform code is located in our case it will be GitHub - tntk-io/tntk-infra but you should you your own repository.

2. Select VCS provider.

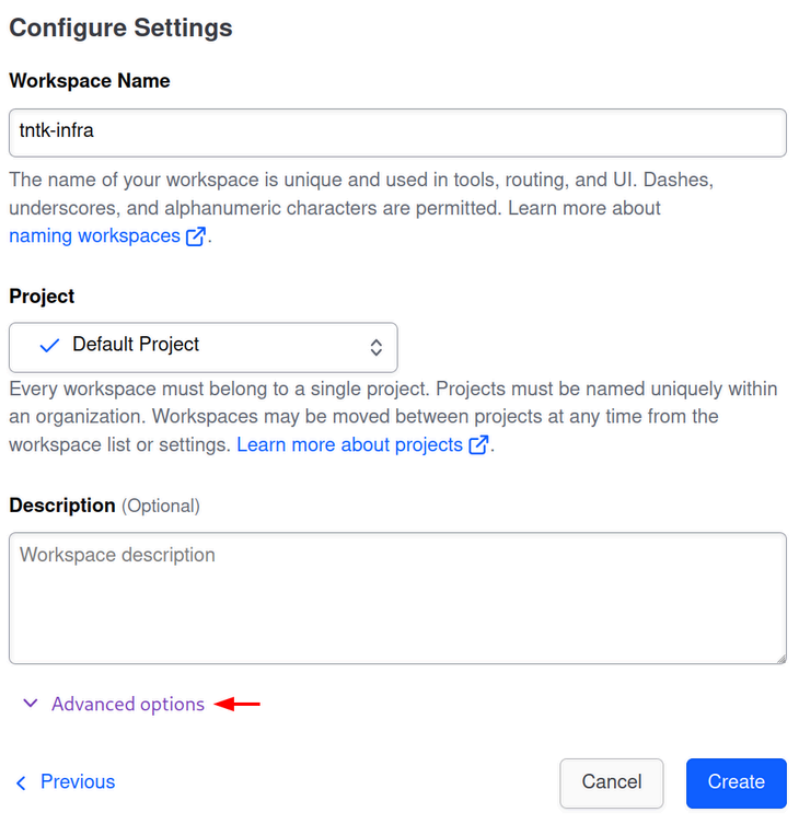


3. Choose repository

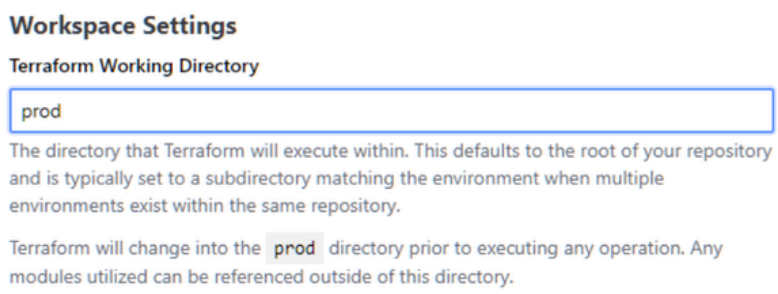


```
structure :
```

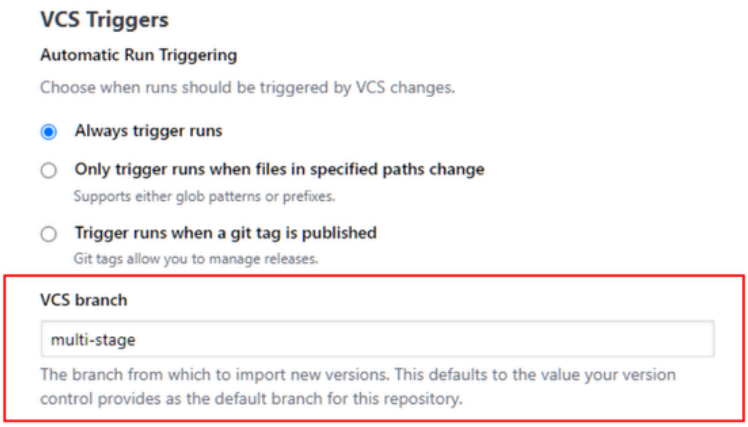
4. Open Advanced options



5. Define folder with terraform code

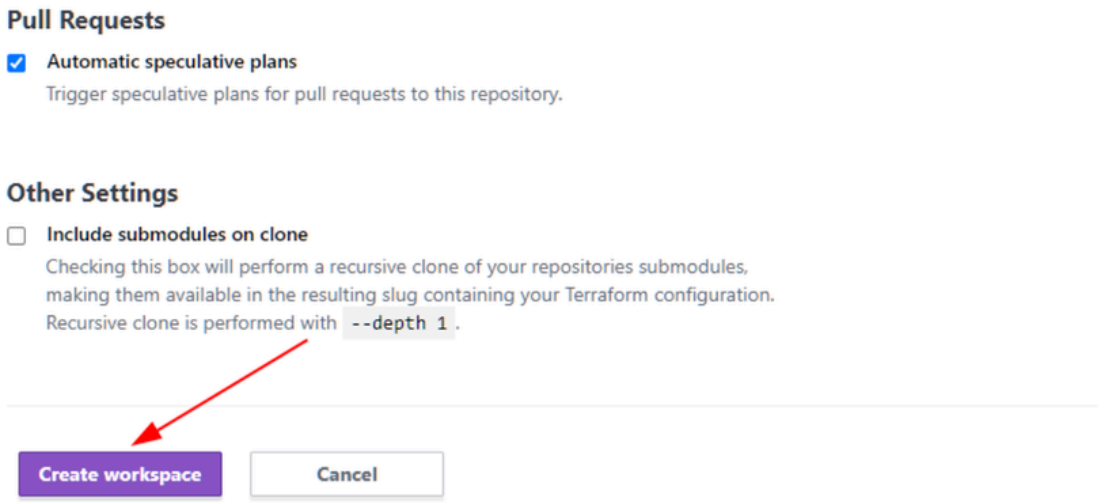


6. Define branch name



or leave it empty if you are using only one default branch

7. Create workspace



8. Define required variables

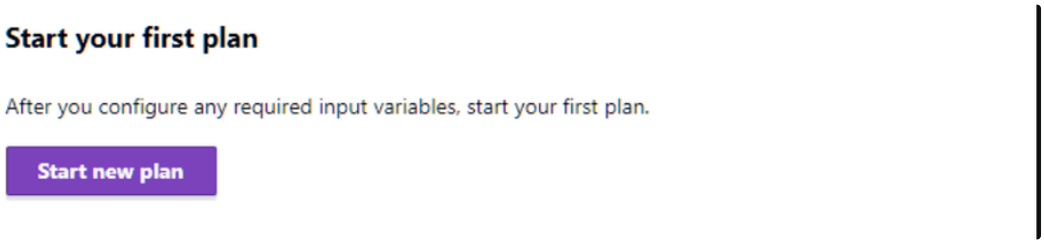
Workspace variables (12)

Variables defined within a workspace always overwrite variables from variable sets that have the same type and the same key. Learn more about variable set precedence.

Key	Value	Category
aws_region	us-east-1	terraform
base_domain	[illegible]	terraform
cd_project_repo	[illegible]	terraform
ci_project_repo	[illegible]	terraform
datadog_api_key SENSITIVE	*Sensitive - write only*	terraform
datadog_application_key SENSITIVE	*Sensitive - write only*	terraform
datadog_region	us5.datadoghq.com	terraform
id_rsa SENSITIVE	*Sensitive - write only*	terraform
registrationToken SENSITIVE	*Sensitive - write only*	terraform
tag_env	prod	terraform
AWS_ACCESS_KEY_ID iam user "admin1" SENSITIVE	*Sensitive - write only*	env
AWS_SECRET_ACCESS_KEY iam user "admin1" SENSITIVE	*Sensitive - write only*	env

Don't forget to define env variables for AWS, if you are using SSO add the AWS_SESSION_TOKEN:

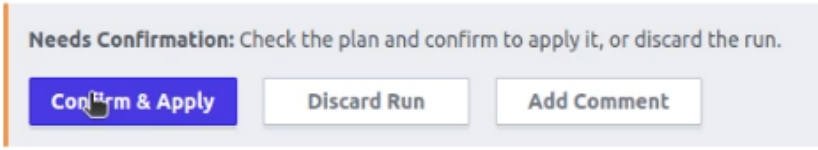
9. Start plan

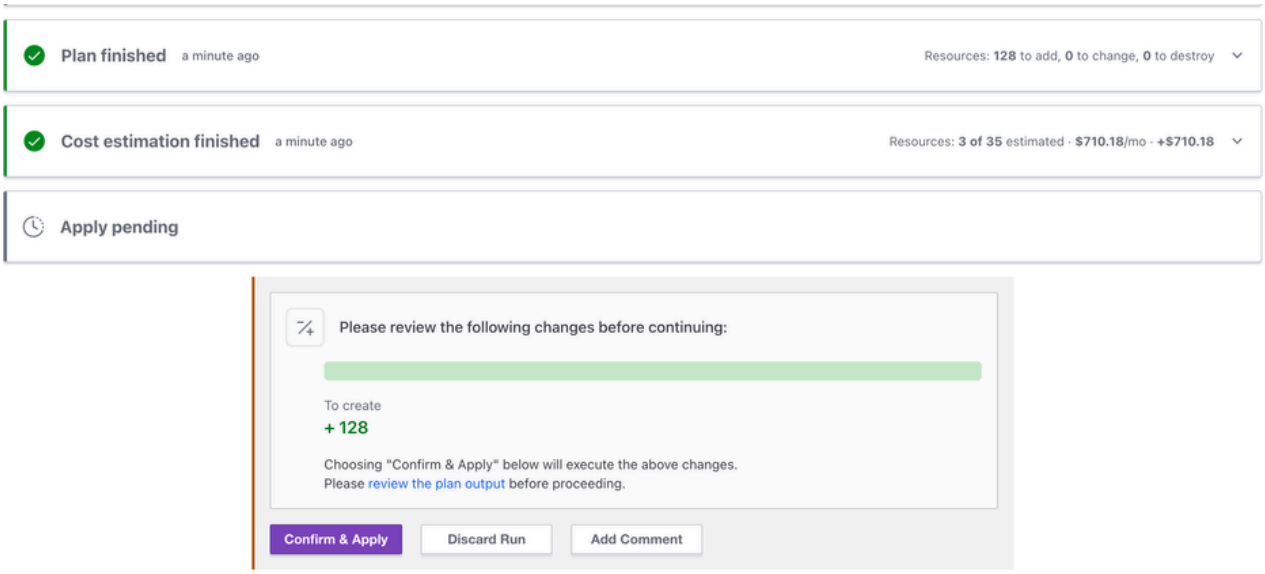


10. Apply

After planning we can **confirm & apply** – it will executes the changes defined by our ***Terraform*** configuration to create, update, or destroy resources. It will take about 30 min to create all resources. **In some cases you could get error on helm creation step, to resolve it you**

just need to start new run and error will goes away





Application deployment

After deploying infrastructure we moving to application section. In this section we will deploy our application to our infastructure.

1. Create secrets and variables

Before starting the CI, you need to create the necessary variables and secrets. You can do this manually through the UI or use the GitHub API REST API endpoints for GitHub Actions Secrets - GitHub Docs. You can also do this via GitHub CLI after performing authentication:

Variables example:

```
gh variable set ACCOUNT_ID --repo tntk-io/tntk-ci --body "012345678901"
gh variable set AWS_REGION --repo tntk-io/tntk-ci --body "us-east-1"
gh variable set BASE_DOMAIN --repo tntk-io/tntk-ci --body "your-domain.com"
gh variable set APPLICATION_NAME --repo tntk-io/tntk-ci --body "demoapp"
gh variable set APPLICATION_NAMESPACE --repo tntk-io/tntk-ci --body "application"
gh variable set CD_DESTINATION_OWNER --repo tntk-io/tntk-ci --body "tntk-io"
gh variable set CD_PROJECT --repo tntk-io/tntk-ci --body "tntk-cd"
```

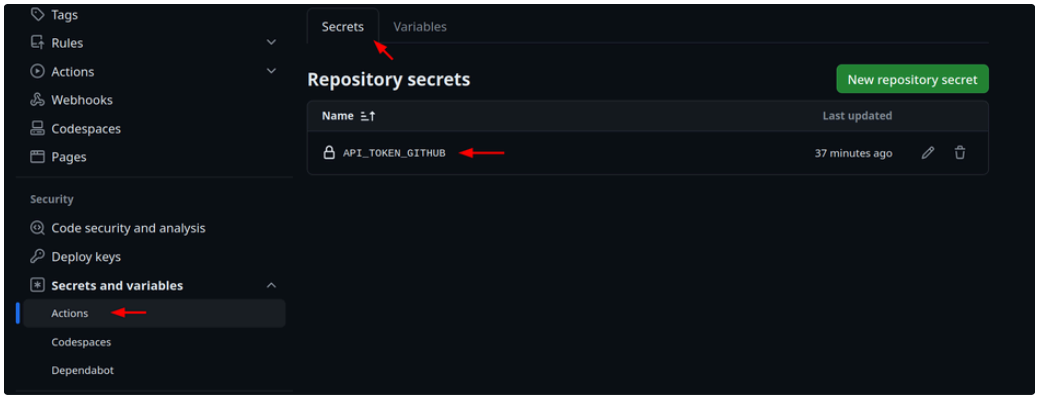
> The data provided in the variables above are for example purposes only. Use your own values.

Also you have to create **secret** named `API_TOKEN_GITHUB` which will be used on the "k8s_manifest_storing" stage:

```
gh secret set API_TOKEN_GITHUB --repo tntk-io/tntk-ci --body "ghp_TEST_SAMPLEdashgfwqohgohr21ihfih12f9hf1"
```

UI example:

Repo > Settings > Secrets and variables > Actions > Secrets/Variables:

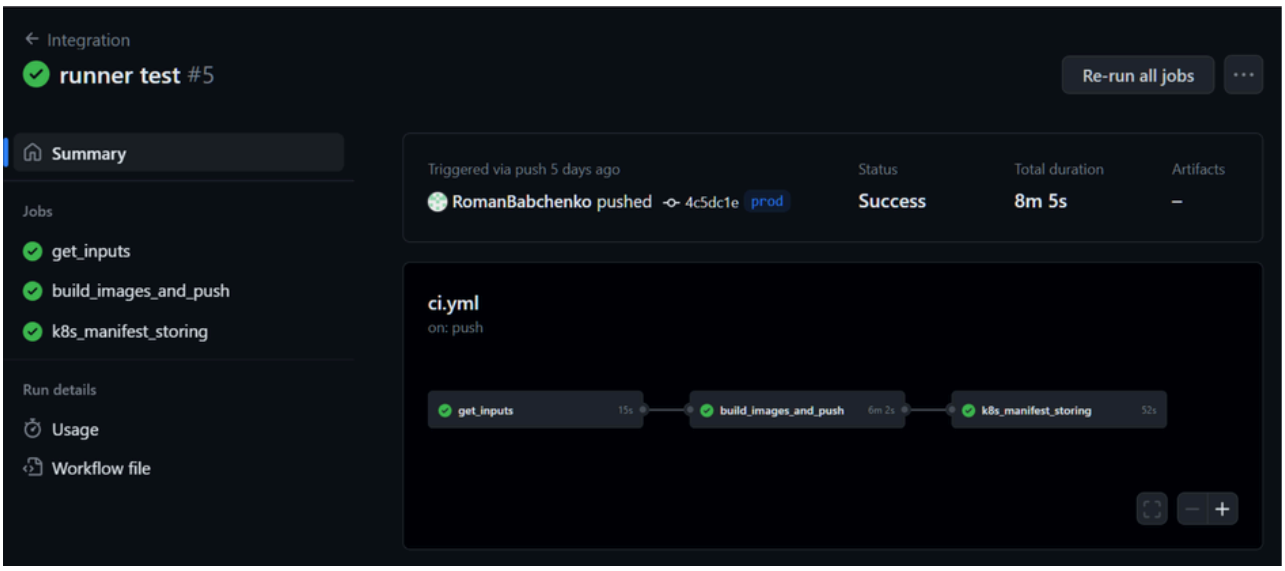


2. Trigger CI pipeline

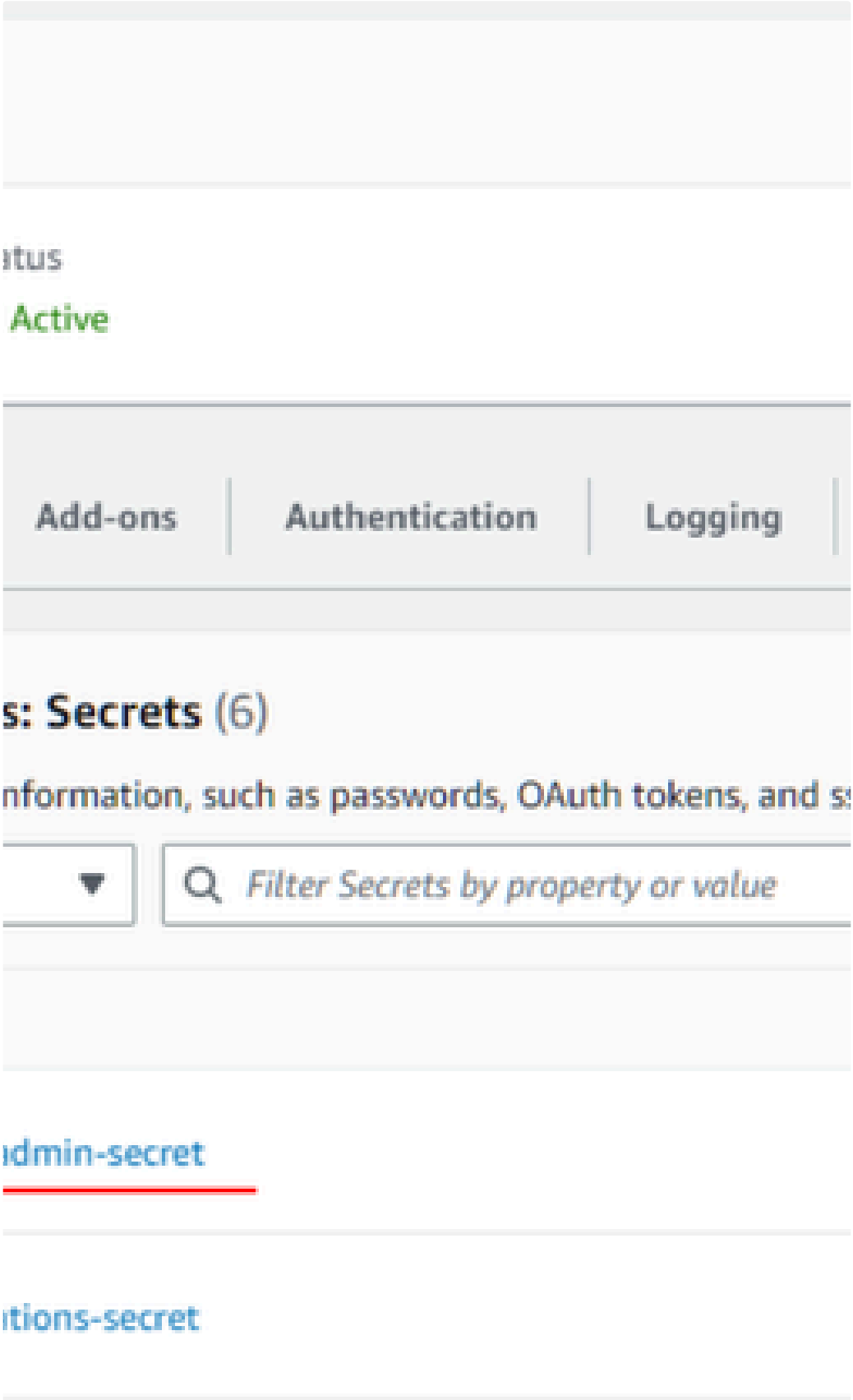
Our integration process depends on commit SHA and the easiest way to start new build is make a empty commit to the CI branch. Trigger CI pipeline by pushing new commit to project src repo:

```
git commit --allow-empty -m "test emplty commit"
```

3. Wait for the CI process to finish



4. Get argocd default admin password



e.v1.argocd-apps.v1

e.v1.argocd.v1

AWS cli:

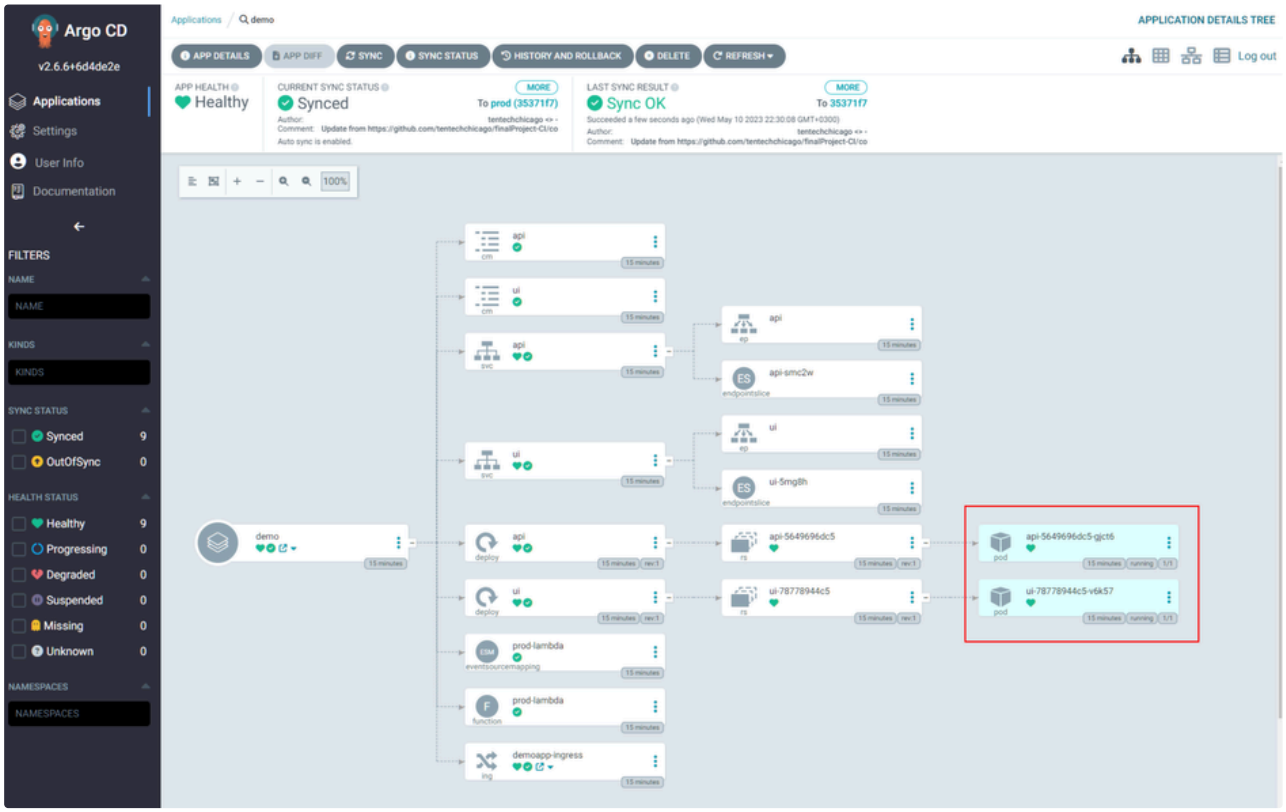
```
aws eks update-kubeconfig --name eks-prod
kubectl -n argocd get secrets argocd-initial-admin-secret -o json | jq -r .data.password | base64 -d
```

5. Login to argo CD and check sync

https://argo.prod.your-domain.com/

login: admin

pass: <your_retrieved_password>



Check the app

https://demoapp.prod.your-domain.com/