GitLab

Empower everyone to manage Infrastructure with GitLab and Terraform

Timo Furrer

GitLab

Timo Furrer

Senior Backend Engineer
@timofurrer

GitLab

Once upon a time ...

What do we want?

- [ ] Low-Friction setup
- [ ] Extend to a “complex” setup
- [ ] Confidence
- [ ] Reusability

Storing State

```
resource "random_pet" "main"
{
  length = 2
}

output "pet_name" {
  value = random_pet.main.id
}
```

Storing State

```
$ terraform init
$ terraform apply
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
$ cat terraform.tfstate
```

Storing State

```
{
  "version": 4,
  "terraform_version": "1.3.6",
  "serial": 2,
  "lineage": "972c2eb1-46ab-e29b-2cda-14baacd0e2fa",
  "outputs": {
    "pet_name": {
      "value": "welcome-antelope",
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "managed",
      "type": "random_pet",
      "name": "main",
      "provider":
"provider[\"registry.terraform.io/hashicorp/random\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "id": "welcome-antelope",
            "keepers": null,
            "length": 2,
            "prefix": null,
            "separator": "-"
          },
          "sensitive_attributes": []
        }
      ]
    }
  ],
  "check_results": null
}
```

Storing State

```
{
  "version": 4,
  "terraform_version": "1.3.6",
  "serial": 2,
  "lineage": "972c2eb1-46ab-e29b-2cda-14baacd0e2fa",
  "outputs": {
    "pet_name": {
      "value": "welcome-antelope",
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "managed",
      "type": "random_pet",
      "name": "main",
      "provider":
"provider[\"registry.terraform.io/hashicorp/random\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "id": "welcome-antelope",
            "keepers": null,
            "length": 2,
            "prefix": null,
            "separator": "-"
          },
          "sensitive_attributes": []
        }
      ]
    }
  ],
  "check_results": null
}
```

Storing State


```
{
  "version": 4,
  "terraform_version": "1.3.6",
  "serial": 2,
  "lineage": "972c2eb1-46ab-e29b-2cda-14baacd0e2fa",
  "outputs": {
    "pet_name": {
      "value": "welcome-antelope",
      "type": "string"
    }
  },
  "resources": [
    {
      "mode": "managed",
      "type": "random_pet",
      "name": "main",
      "provider":
"provider[\"registry.terraform.io/hashicorp/random\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "id": "welcome-antelope",
            "keepers": null,
            "length": 2,
            "prefix": null,
            "separator": "-"
          },
          "sensitive_attributes": []
        }
      ]
    }
  ],
  "check_results": null
}
```

Storing State

Let's store the state remotely:

```
terraform {
  backend "http" {}
}

resource "random_pet" "main"
{
  length = 2
}

output "pet_name" {
  value = random_pet.main.id
}
```

Storing State

We need to configure the backend:

```
terraform {
  backend "http" {
    // address?
    // (un)lock address?
    // username & password?
    // timeouts?
  }
}

resource "random_pet" "main"
{
  ..
}
```

GitLab-managed state

We can configure it manually in the config:

```
terraform {
  backend "http" {
    address        = "https://gitlab.com/…/state-name"
    lock_address   =
"https://gitlab.com/…/state-name/lock"
    unlock_address =
"https://gitlab.com/…/state-name/lock"
    username       = "__token__"
    password       = "glpat-..."
    lock_method    = "POST"
    unlock_method  = "DELETE"
    ...
  }
}

resource "random_pet" "main"  {
  ...
}
```

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/terraform_state.html

GitLab-managed state

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/terraform_state.html

... or in the terminal during initialization:

```
PROJECT_ID="<gitlab-project-id>"
TF_USERNAME="<gitlab-username>"
TF_PASSWORD="<gitlab-personal-access-token>"
TF_ADDRESS="https://gitlab.com/api/v4/projects/${PROJECT_ID}/terraform/state/state-name"

terraform init \
  -backend-config=address=${TF_ADDRESS} \
  -backend-config=lock_address=${TF_ADDRESS}/lock \
  -backend-config=unlock_address=${TF_ADDRESS}/lock \
  -backend-config=username=${TF_USERNAME} \
  -backend-config=password=${TF_PASSWORD} \
  -backend-config=lock_method=POST \
  -backend-config=unlock_method=DELETE \
  -backend-config=retry_wait_min=5
```

GitLab

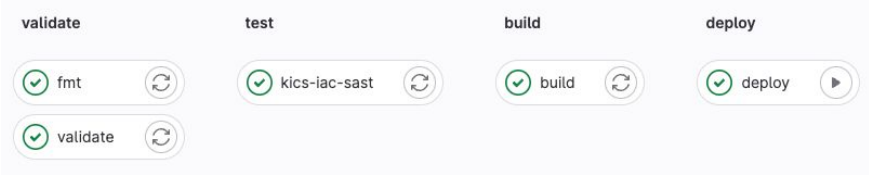
Terraform CI/CD Templates

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/

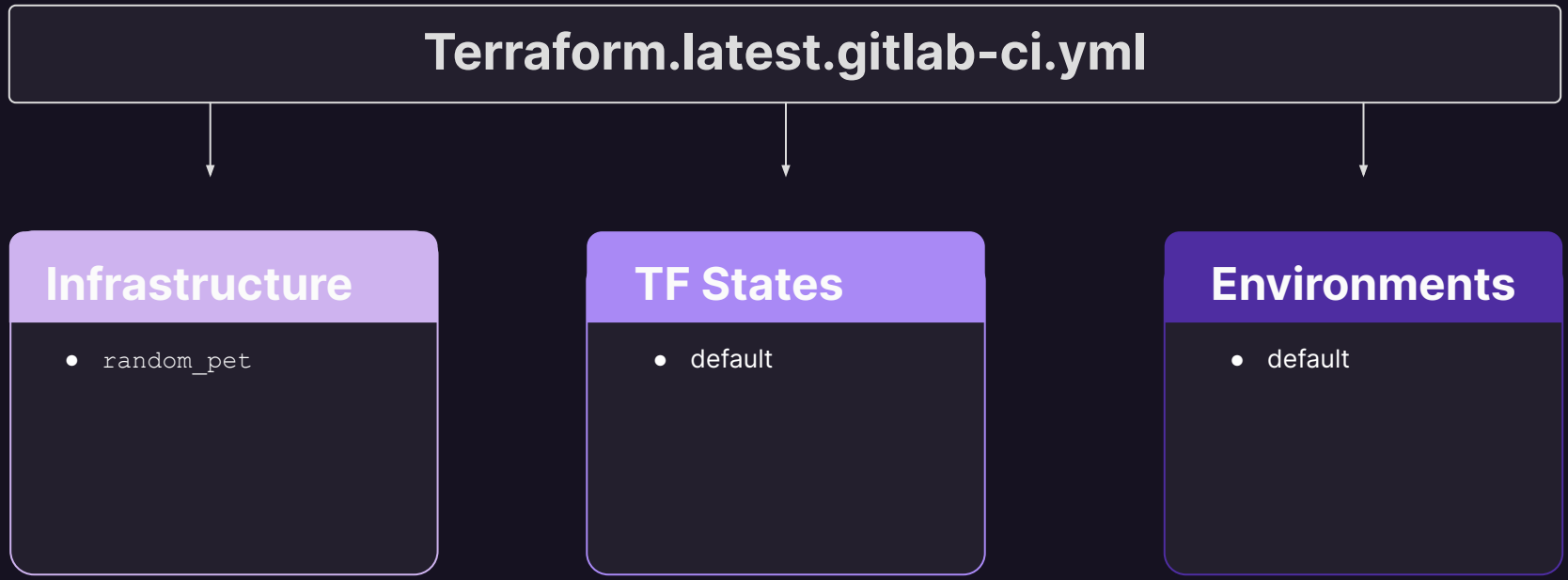
Pipeline

Include the template in the `.gitlab-ci.yml`:

```
include:
  template: Terraform.latest.gitlab-ci.yml
```



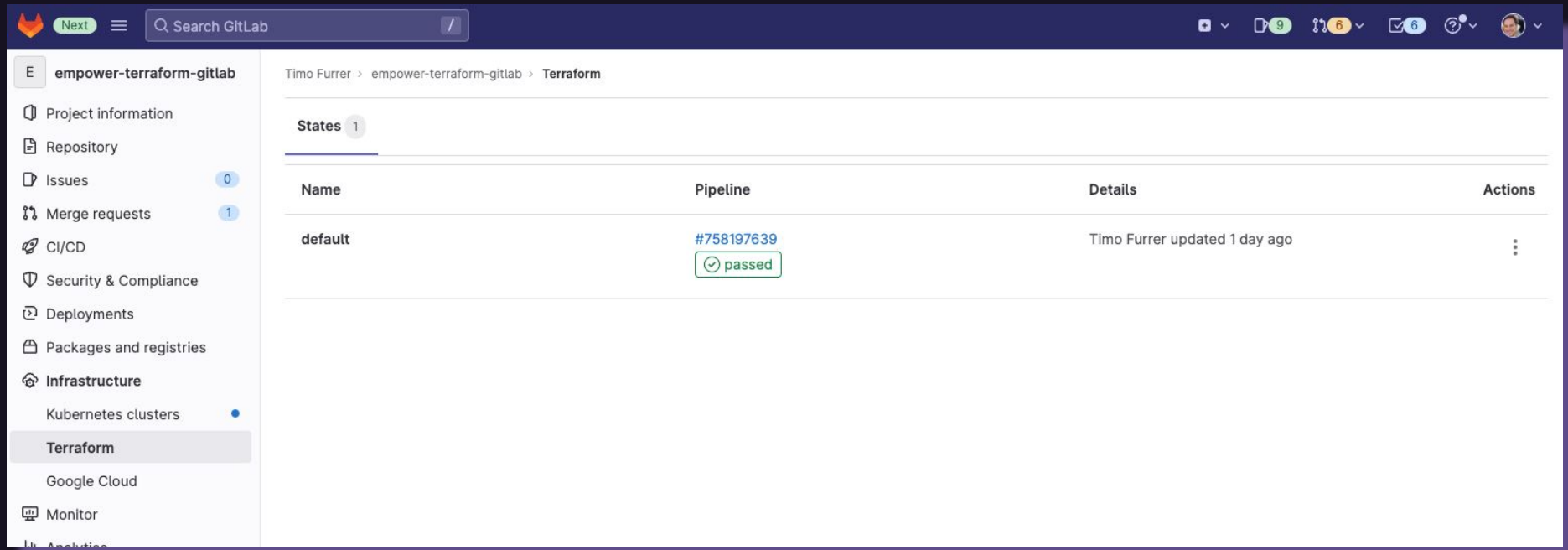
GitLab


Terraform.latest.gitlab-ci.yml
Infrastructure
random_pet
TF States
default
Environments
default

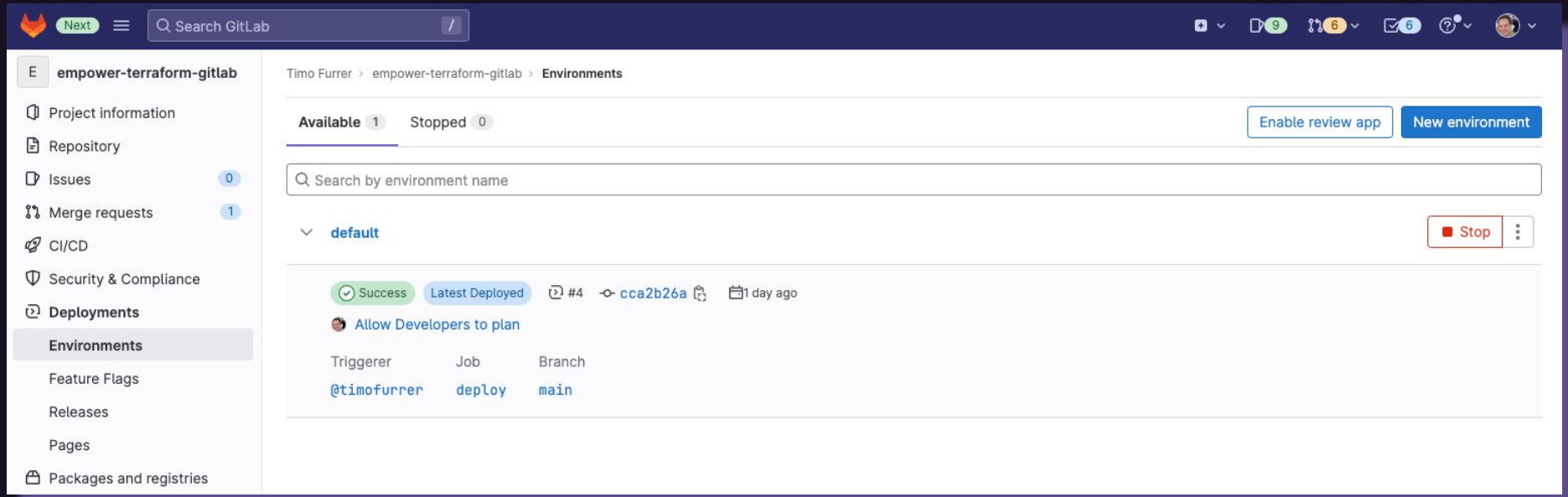
GitLab

Show me! ...

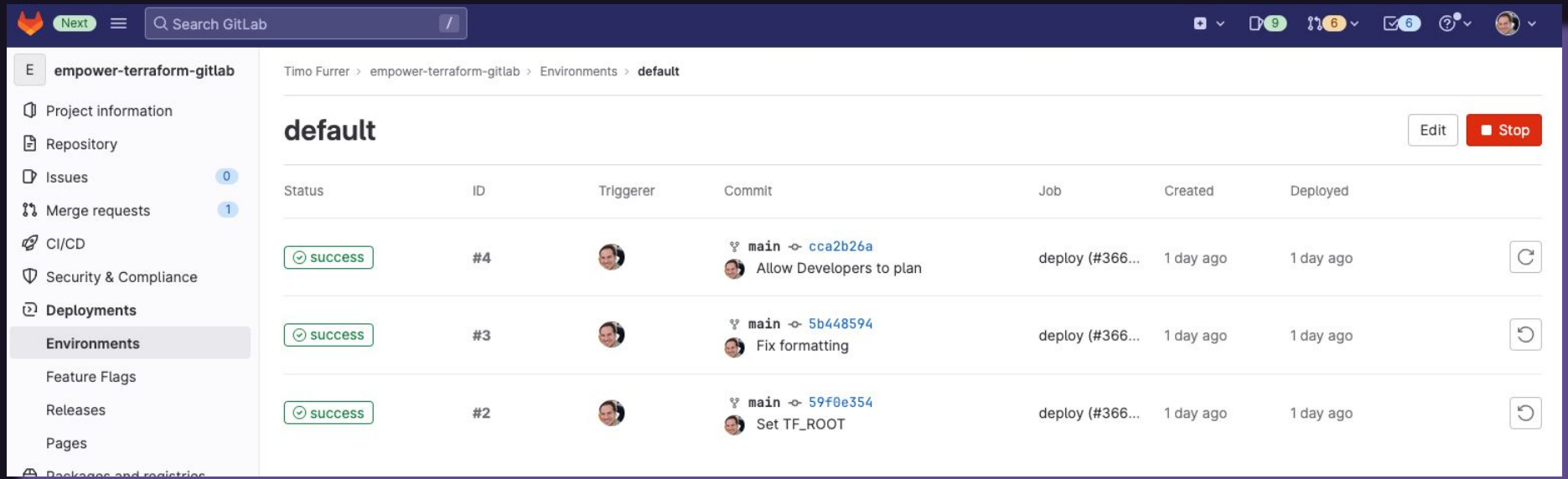
Terraform State



Environment



Environment Deployments



What do we want?

- [x] Low-Friction setup
- [ ] Extend to a “complex” setup
- [ ] Confidence
- [ ] Reusability

Customize Pipeline

Give the state and environment a name:

```
include:
  template: Terraform.latest.gitlab-ci.yml

variables:
  TF_STATE_NAME: production
```

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Customize Pipeline

Place the Terraform config in a directory:

```
include:
  template: Terraform.latest.gitlab-ci.yml

variables:
  TF_STATE_NAME: production
  TF_ROOT: terraform/
```

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Extend Template Jobs

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/terraform_template_recipes.html

Setup Vault Authentication:

```
include:
  template: Terraform.latest.gitlab-ci.yml

.vault-auth: &vault_auth
  - apk add --no-cache vault
  - export VAULT_ADDR=http://vault.example.com:8200
  - export VAULT_TOKEN="$(
      vault write -field=token auth/jwt/login
      role=$CI_PROJECT_NAME jwt=$CI_JOB_JWT
    )"

build:
  before_script:
    - *vault_auth

deploy:
  before_script:
    - *vault_auth
```

Extend Template Jobs

Setup Vault Authentication:

```
include:
  template: Terraform.latest.gitlab-ci.yml

.vault-auth: &vault_auth
  - apk add --no-cache vault
  - export VAULT_ADDR=http://vault.example.com:8200
  - export VAULT_TOKEN="$(
      vault write -field=token auth/jwt/login
      role=$CI_PROJECT_NAME jwt=$CI_JOB_JWT
    )"

build:  # override job from the template
  before_script:
    - *vault_auth

deploy:  # override job from the template
  before_script:
    - *vault_auth
```

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/terraform_template_recipes.html

Extend Template Jobs

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/terraform_template_recipes.html

Setup Vault Authentication:

```
include:
  template: Terraform.latest.gitlab-ci.yml

.vault-auth: &vault_auth  # Keep it DRY
  - apk add --no-cache vault
  - export VAULT_ADDR=http://vault.example.com:8200
  - export VAULT_TOKEN="$(
      vault write -field=token auth/jwt/login
      role=$CI_PROJECT_NAME jwt=$CI_JOB_JWT
    )"

build:  # override job from the template
  before_script:
    - *vault_auth

deploy:  # override job from the template
  before_script:
    - *vault_auth
```

Multi Environment

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Extend to multiple environments:

```
.environment:
  stage: environments
  variables:
    TF_STATE_NAME: $environment
    TF_CLI_ARGS_plan:
"-var-file=environments/$TF_STATE_NAME.tfvars"
  trigger:
    include:
      template: Terraform.latest.gitlab-ci.yml
    strategy: depend

staging:
  extends: .environment
  variables:
    environment: staging

production:
  extends: .environment
  variables:
    environment: production
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

Multi Environment

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Extend to multiple environments:

```
.environment:
  stage: environments
  variables:
    TF_STATE_NAME: $environment
    TF_CLI_ARGS_plan: "-var-file=environments/$TF_STATE_NAME.tfvars"
  trigger:
    include:
      template: Terraform.latest.gitlab-ci.yml
    strategy: depend

staging:  # Job to trigger `staging` environment
  extends: .environment
  variables:
    environment: staging

production:  # Job to trigger `production` environment
  extends: .environment
  variables:
    environment: production
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

Multi Environment

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Extend to multiple environments:

```
.environment:
  stage: environments
  variables:
    TF_STATE_NAME: $environment
    TF_CLI_ARGS_plan: "-var-file=environments/$TF_STATE_NAME.tfvars"
  trigger:
    include:
      template: Terraform.latest.gitlab-ci.yml
    strategy: depend

staging:  # Job to trigger `staging` environment
  extends: .environment
  variables:
    environment: staging

production:  # Job to trigger `production` environment
  extends: .environment
  variables:
    environment: production
  rules:
    # only for default branch
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

Multi Environment

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Extend to multiple environments:

```
.environment:
  stage: environments
  variables:
    TF_STATE_NAME: $environment
    TF_CLI_ARGS_plan: "-var-file=environments/$TF_STATE_NAME.tfvars"
  trigger:
    include:
      template: Terraform.latest.gitlab-ci.yml
    strategy: depend

staging:  # Job to trigger `staging` environment
  extends: .environment
  variables:
    environment: staging

production:  # Job to trigger `production` environment
  extends: .environment
  variables:
    environment: production
  rules:
    # only for default branch
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

Multi Environment

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Extend to multiple environments:

```
.environment:  # base job template
  stage: environments
  variables:
    TF_STATE_NAME: $environment
    TF_CLI_ARGS_plan: "-var-file=environments/$TF_STATE_NAME.tfvars"
  trigger:
    include:
      template: Terraform.latest.gitlab-ci.yml
    strategy: depend

staging:  # Job to trigger `staging` environment
  extends: .environment
  variables:
    environment: staging

production:  # Job to trigger `production` environment
  extends: .environment
  variables:
    environment: production
  rules:
    # only for default branch
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

Multi Environment

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Extend to multiple environments:

```
.environment:  # base job template
  stage: environments
  variables:
    # Environment specific variables:
    TF_STATE_NAME: $environment
    TF_CLI_ARGS_plan: "-var-file=environments/$TF_STATE_NAME.tfvars"
  trigger:
    include:
      template: Terraform.latest.gitlab-ci.yml
    strategy: depend

staging:  # Job to trigger `staging` environment
  extends: .environment
  variables:
    environment: staging

production:  # Job to trigger `production` environment
  extends: .environment
  variables:
    environment: production
  rules:
    # only for default branch
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

Multi Environment

Docs: https://docs.gitlab.com/ee/user/infrastructure/iac/gitlab_terraform_helpers.html

Extend to multiple environments:

```
.environment:  # base job template
  stage: environments
  variables:
    # Environment specific variables:
    TF_STATE_NAME: $environment
    TF_CLI_ARGS_plan: "-var-file=environments/$TF_STATE_NAME.tfvars"
  trigger:  # trigger child-pipeline
    include:
      template: Terraform.latest.gitlab-ci.yml  # include the template
    strategy: depend

staging:  # Job to trigger `staging` environment
  extends: .environment
  variables:
    environment: staging

production:  # Job to trigger `production` environment
  extends: .environment
  variables:
    environment: production
  rules:
    # only for default branch
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
```

GitLab

Child Pipeline: Terraform.latest.gitlab-ci.yml

Infrastructure

- `random_pet [len=3]`
- `random_pet [len=2]`

TF States

- staging
- production

Environments

- staging
- production

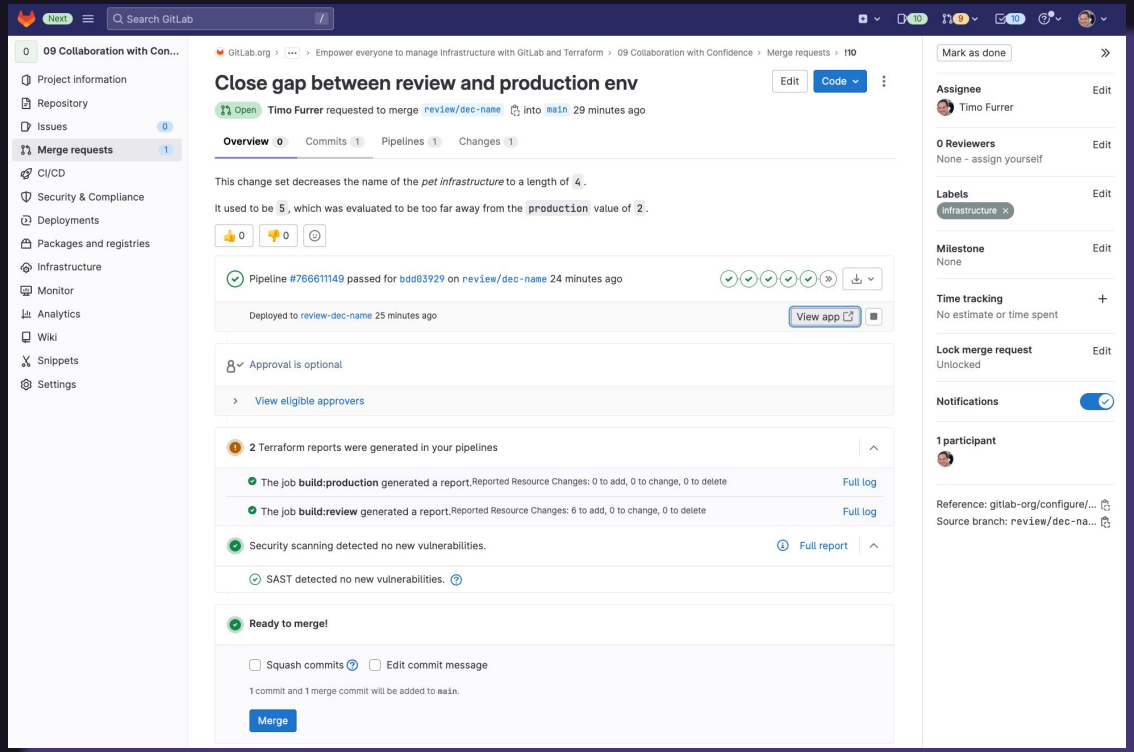
What do we want?

- [x] Low-Friction setup
- [x] Extend to a “complex” setup
- [ ] Confidence
- [ ] Reusability

Confidence

- Use Multiple Environments
- Use Review Apps
- Use Merge Request Terraform Widget
- Use static code analysis report
- Run tests
  - simple curl commands ¯_(ツ)_/¯
  - Terratest
  - ...

Collaboration



[Demo]

Permissions

Who has access to the state?

	Maintainers	Developers
read	✓	✓
write	✓	
lock	✓	
unlock	✓	

Permissions

Use state-only project to better control state access:

```
include:
  template: Terraform.latest.gitlab-ci.yml

variables:
  TF_STATE_NAME: production
  TF_PROJECT_ID: 123456
  # define TF_STATE_ADDRESS
  TF_STATE_ADDRESS:
$CI_API_V4_URL/projects/$TF_PROJECT_ID/ter
raform/state/${TF_STATE_NAME}}
```

Open Issue: Protected Terraform States

What do we want?

- [x] Low-Friction setup
- [x] Extend to a “complex” setup
- [x] Confidence
- [ ] Reusability

GitLab

Sharing Modules

Docs: https://docs.gitlab.com/ee/user/packages/terraform_module_registry/index.html

Modules

What is a Module again?

“A Terraform module is a set of Terraform configuration files in a single directory.”

- Terraform Docs

```
$ ls pet-site/
.
├── LICENSE
├── README.md
├── main.tf
├── variables.tf
├── outputs.tf
```

Modules

We can use a Terraform Module:

```
$ cat production/main.tf

module "pet_site" {
  source  = "../pet-site"

  pet_name_length  = 2
}
```

```
$ cat staging/main.tf

module "pet_site" {
  source  = "../pet-site"

  pet_name_length  = 4
}
```

Modules

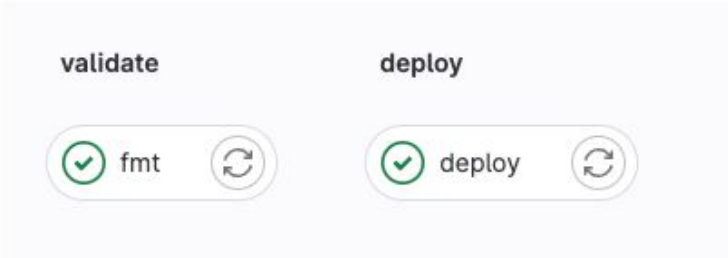
We can use a remote Terraform Module:

```
module "pet_site" {
  source  = "gitlab.com/gitlab-org/pet-site/aws"
  version = "~> 1.0"

  pet_name_length = 2
}
```

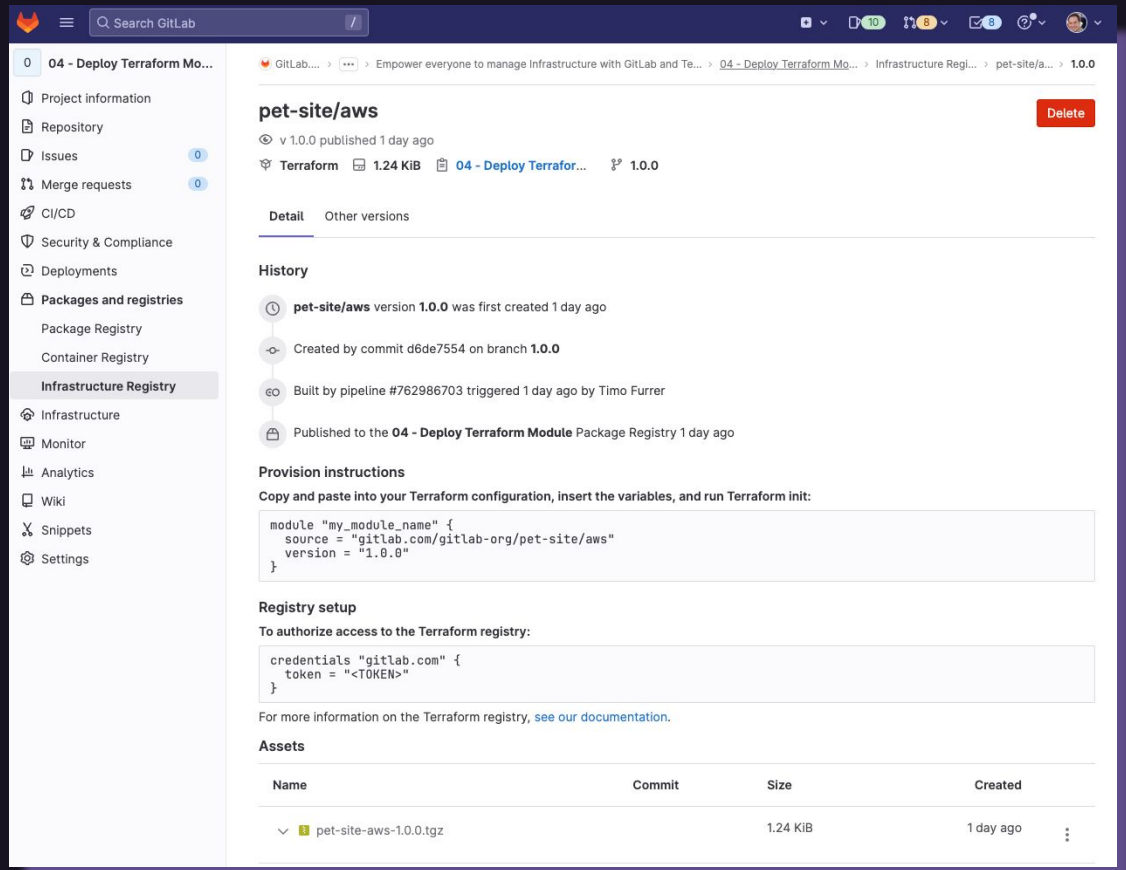
Publish Module

Include module template in `.gitlab-ci.yml`:

```
include:
  template: Terraform-Module.gitlab-ci.yml

variables:
  TERRAFORM_MODULE_DIR: ${CI_PROJECT_DIR}
  TERRAFORM_MODULE_NAME: ${CI_PROJECT_NAME}
  TERRAFORM_MODULE_SYSTEM: local
  TERRAFORM_MODULE_VERSION:${CI_COMMIT_TAG}
```



Infrastructure Registry



What do we want?

- [x] Low-Friction setup
- [x] Extend to a “complex” setup
- [x] Confidence
- [x] Reusability

GitLab

One more thing ...

Docs: https://registry.terraform.io/providers/gitlabhq/gitlab/latest/docs

GitLab Provider

```
terraform {
  required_providers {
    gitlab = {
      source  = "gitlabhq/gitlab"
      version = "~> 15.8"
    }
  }
}

provider "gitlab" {
  # Configuration options
}
```

GitLab Provider

```
terraform {
  required_providers {
    gitlab = {
      source  = "gitlabhq/gitlab"
      version = "~> 15.8"
    }
  }
}

provider "gitlab" {
  # Configuration options
  base_url = "gitlab.example.com"
  token    = var.gitlab_token
}
```

67+ Resources
40+ Data Sources

```
terraform {
  required_providers {
    gitlab = {
      source  = "gitlabhq/gitlab"
      version = "~> 15.8"
    }
  }
}

provider "gitlab" {
  # Configuration options
}
```

Popular Use Cases

- **User onboarding**

```
data "gitlab_group" "backend_engineers" {
  full_path = "roles/backend-engineers"
}

resource "gitlab_user" "jessica_drew" {
  name     = "Jessica Drew"
  username = "jessicadrew"
  email    = "jessica.drew@marvel.example.com"
}

resource "gitlab_group_membership" "jessica_drew_backend" {
  user_id      = gitlab_user.jessica_drew.id
  group_id     = data.gitlab_group.backend_engineers.id
  access_level = "maintainer"
}
```

Popular Use Cases

- User onboarding
- **(compliant) group / project creation**

  - Protected Branch, Tag and Environment settings
  - Standardized permission settings
  - Approval rules
  - CODEOWNER settings
  - Integrations (slack, ...)
  - CI/CD variables
  - Issue Boards
  - ...

Popular Use Cases

- User onboarding
- (compliant) group / project creation
- **Instance management**

  - Application Settings
  - Topic management
  - System Hooks
  - ...

Popular Use Cases

- User onboarding
- (compliant) group / project creation
- Instance management
- **Runner management**

```
resource "gitlab_group" "my_group" {
  name        = "my-group"
  description = "Group that holds the runners"
}

resource "gitlab_runner" "main" {
  registration_token = gitlab_group.my_group.runners_token
}
```

Popular Use Cases

- User onboarding
- (compliant) group / project creation
- Instance management
- Runner management
- **GitLab Agent for Kubernetes management**

```
resource "gitlab_cluster_agent" "this"{
  project = "prd-cluster"
  name    = "my-agent"
}

resource "gitlab_cluster_agent_token" "this"{
  project     = gitlab_cluster_agent.this.project
  agent_id    = gitlab_cluster_agent.this.agent_id
  name        = "prd"
}

resource "helm_release" "gitlab_agent"{
  name             = "gitlab-agent"
  namespace        = "gitlab-agent"
  create_namespace = true
  repository       = "https://charts.gitlab.io"
  chart            = "gitlab-agent"
  version          = "1.10.0"

  set {
    name  = "config.token"
    value = gitlab_cluster_agent_token.this.token
  }
}
```

GitLab

Thank you!

GitLab

Timo Furrer

Senior Backend Engineer
@timofurrer


GitLab