Automate Your Infrastructure with GitOps, Terraform and Drone

Image from Europeana on Unsplash

Meltwater

Hello, I’m Jim Sheldon!

Nice to meet you :)

- Infrastructure Engineer @ Meltwater for 5 years
- I live in New Hampshire, USA with my wife and our three children

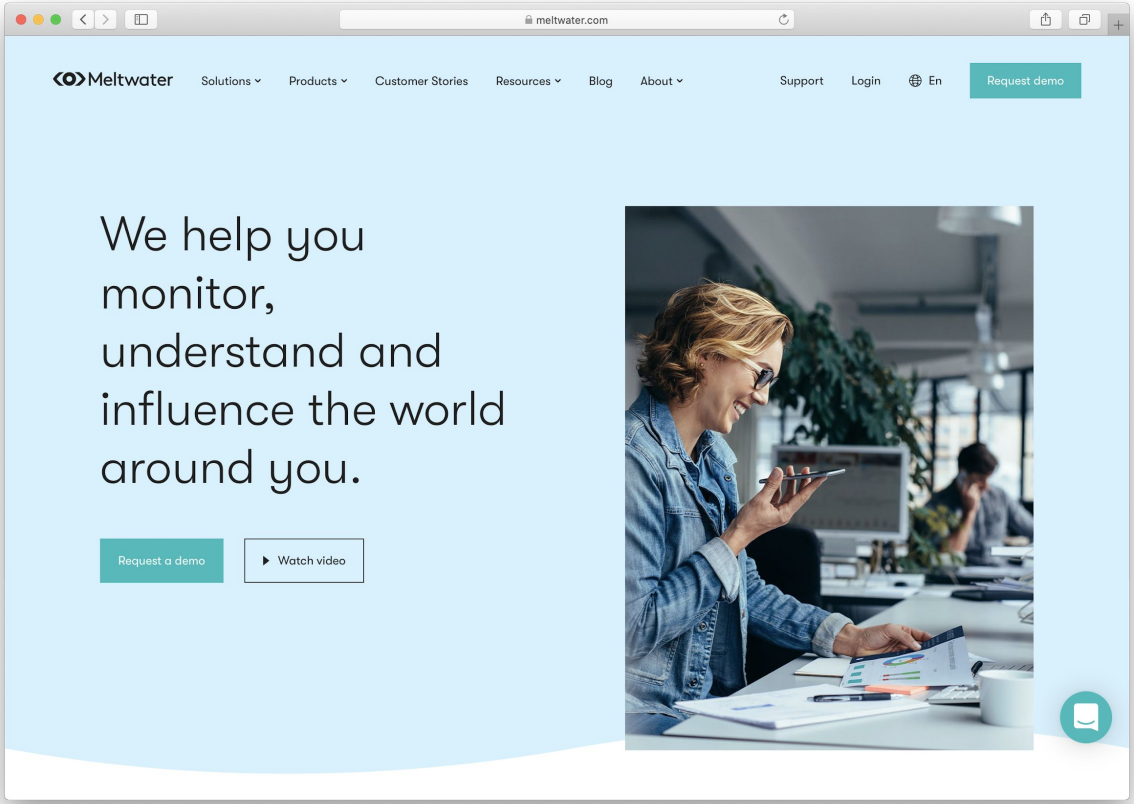
@bitberk

jimsheldon

“Deliver the leading integrated media and social intelligence solution for PR, Comms and Marketing departments”

Meltwater

https://meltwater.com



Meltwater

Meltwater Products

Media Monitoring

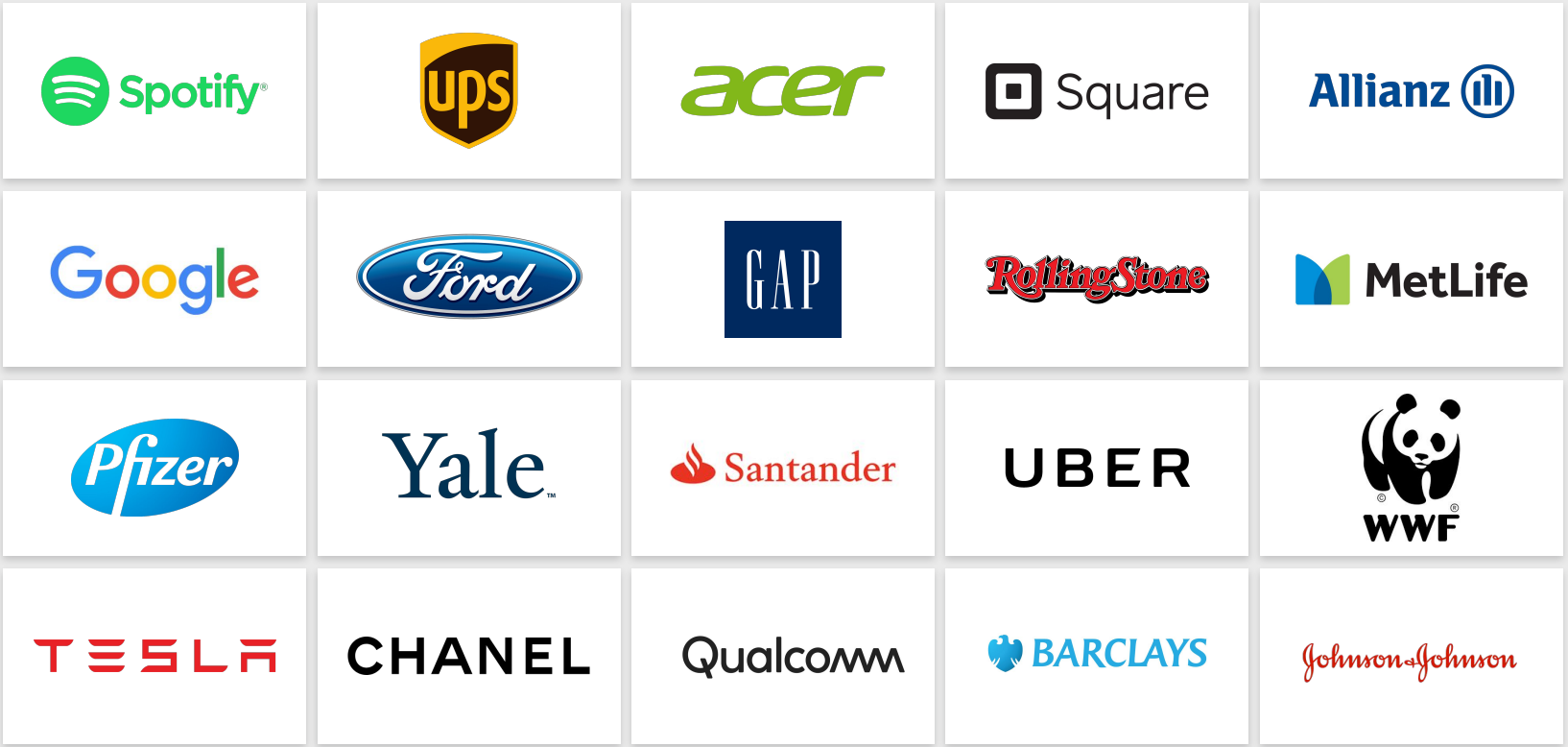
Media Outreach

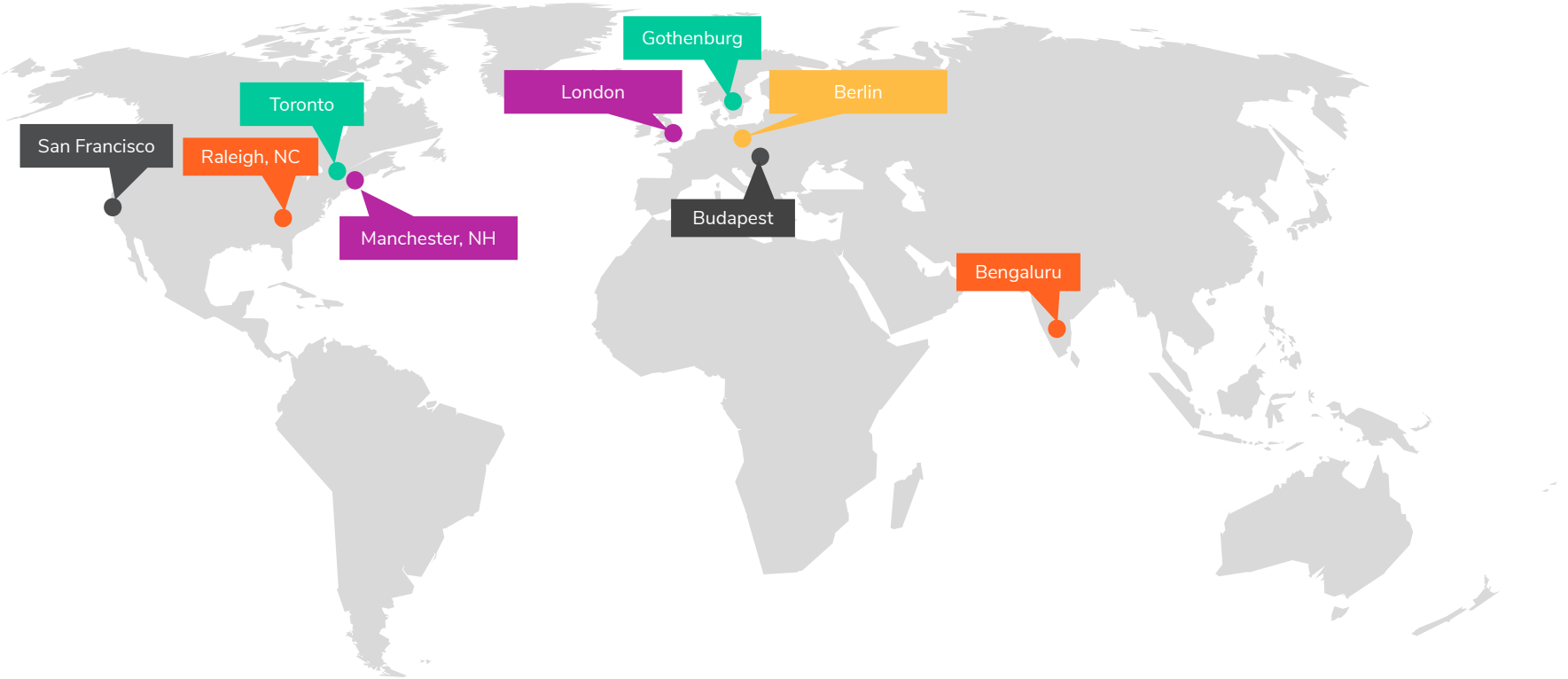
Social Listening

Social Publishing & Engagement

Social Influencer Management

Some of our 30K+ clients





~300 Engineers 9 Offices

Meltwater

FOUNDATION

“

“We enable teams to **accelerate** without thinking too much about the infrastructure, allowing a rapid path from **ideation** to **prototype** to providing **business value**”

What are we talking about?

GitOps

Git is the “source of truth” for all running infrastructure

Terraform

Infrastructure-as-code of choice

Drone

CI/CD solution

Why GitOps?

- Follow proven software development processes, Infrastructure-as-code *is code!*
- Easily see differences over time
- Pull requests checks let us know what is changing, and give us an audit trail to refer back to
- Easily roll back changes by reverting commits
- Our GitHub organization is a searchable resource

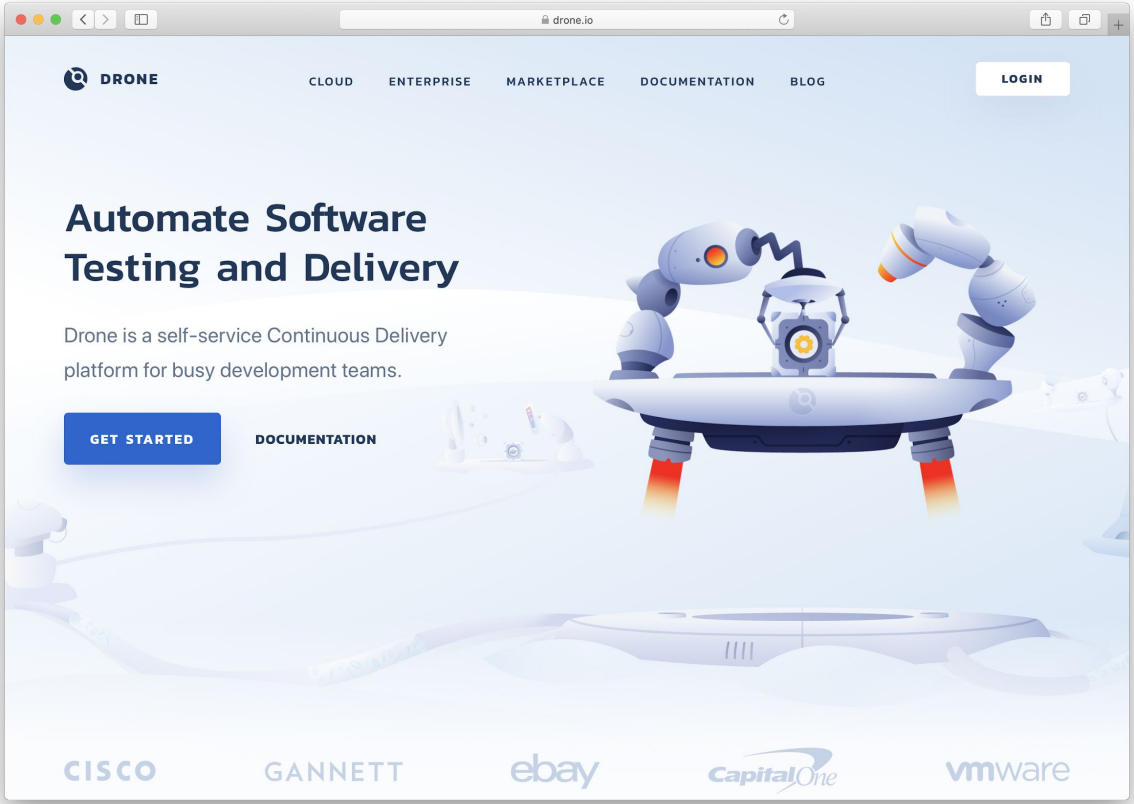
Why Terraform?

- Cloud agnostic
- Dozens of providers (AWS, Datadog, Kubernetes, etc)
- Idempotent
- Excellent documentation
- Fairly low learning curve

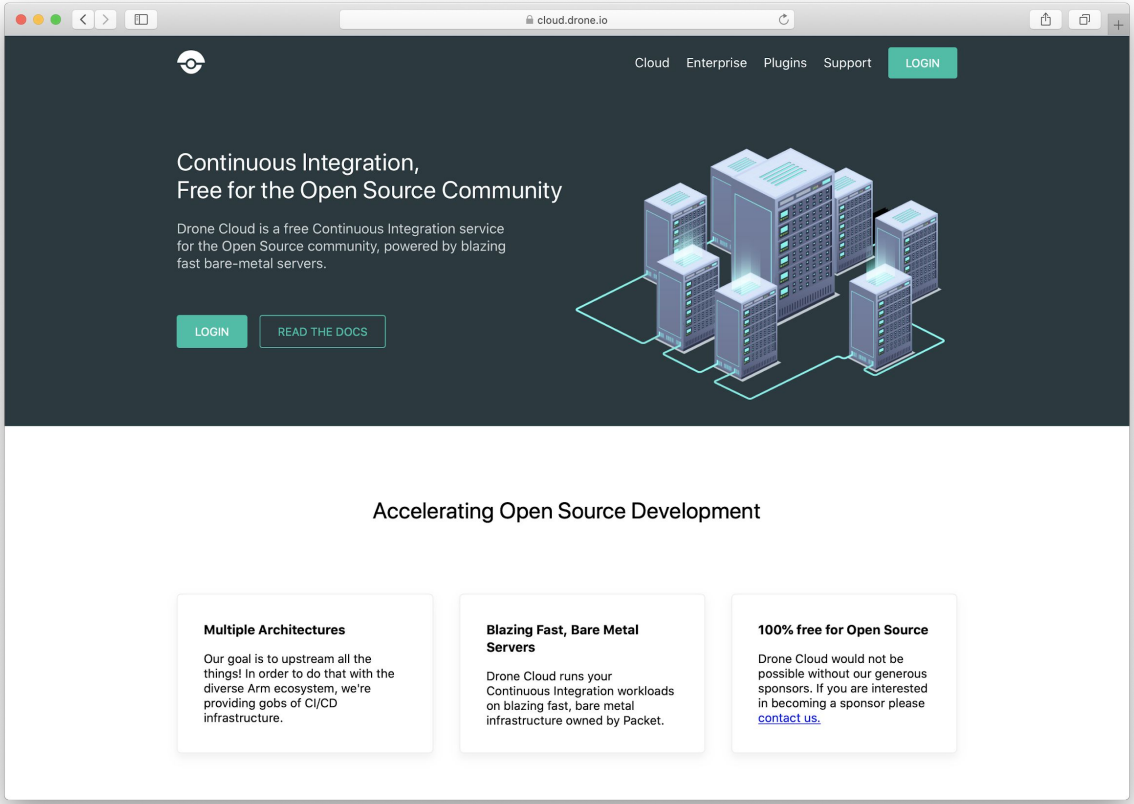
Why Drone?

- Container-native
- Unopinionated
- Large ecosystem of plugins
- GitHub for authentication
- Ease of administration
- Works for just about everything (Terraform, Kubernetes…)

https://drone.io



https://cloud.drone.io



Our Internal Drone Service

1,382 repos

227 users

1,000+ / day

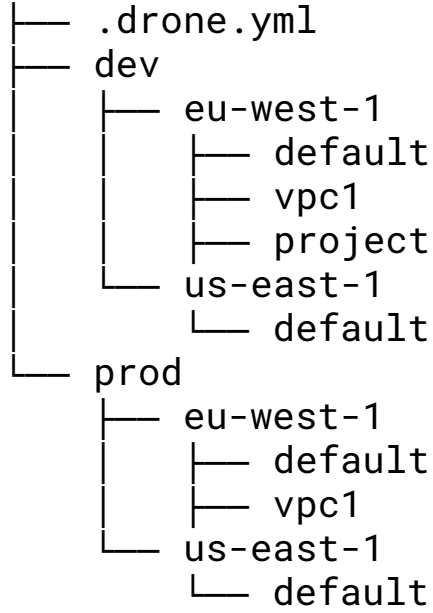
28

Meltwater Terraform repositories with Drone pipelines run in the past 30 days

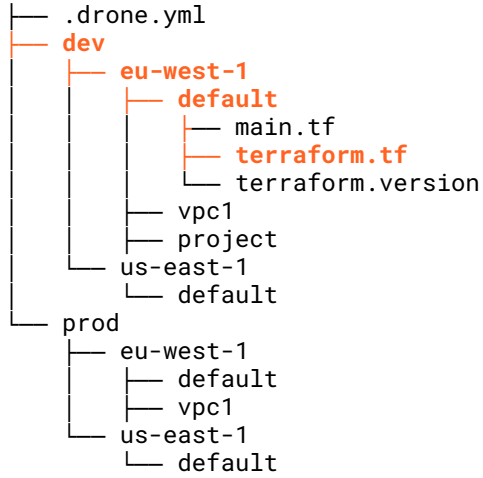
448

Individual Meltwater Terraform pipelines run in the past 30 days

Repository directory structure



State files

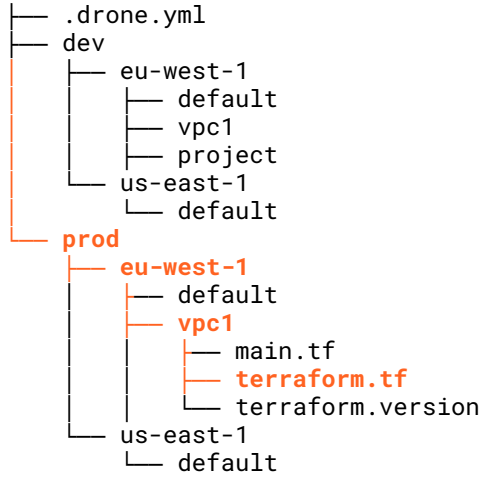


terraform.tf

```
terraform {
  backend "s3" {
    bucket = "tf-state-mw-myteam-dev"
    key    = "eu-west-1/default.tfstate"
    region = "eu-west-1"
  }
}
```

Meltwater

State files



terraform.tf

```
terraform {
  backend "s3" {
    bucket = "tf-state-mw-myteam-prod"
    key    = "eu-west-1/vpc1.tfstate"
    region = "eu-west-1"
  }
}
```

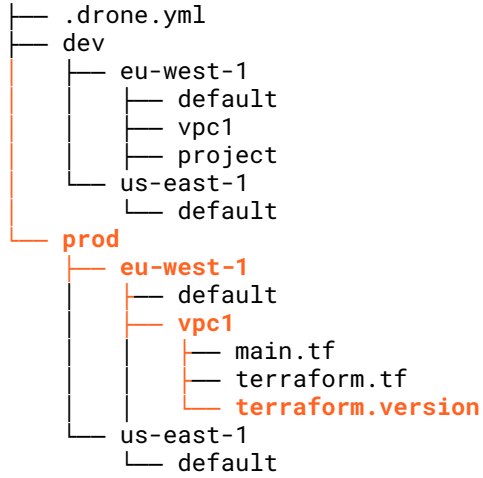
Terraform version files

```
├── .drone.yml
├── dev
│   ├── eu-west-1
│   │   ├── default
│   │   │   ├── main.tf
│   │   │   ├── terraform.tf
│   │   │   └── terraform.version
│   │   ├── vpc1
│   │   └── project
│   └── us-east-1
│       └── default
└── prod
    ├── eu-west-1
    │   ├── default
    │   └── vpc1
    └── us-east-1
        └── default
```

terraform.version

```
0.12.14
```

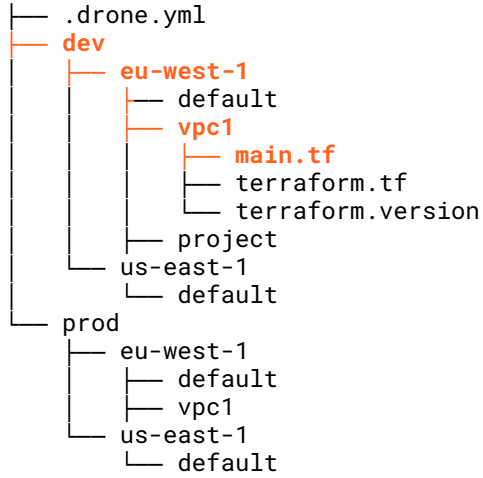
Terraform version files



terraform.version

```
0.11.14
```

Modules



main.tf

```
module "example" {
  source =
"git::git@github.com:meltwater/terraform-aws-
modules.git//example?ref=example_v2"

  foo = "bar"
}
```

Meltwater

Modules

```
├── .drone.yml
├── dev
│   ├── eu-west-1
│   │   ├── default
│   │   ├── vpc1
│   │   └── project
│   └── us-east-1
│       └── default
└── prod
    ├── eu-west-1
    │   ├── default
    │   ├── vpc1
    │   │   ├── main.tf
    │   │   ├── terraform.tf
    │   │   └── terraform.version
    └── us-east-1
        └── default
```

main.tf

```
module "example" {
  source =
"git::git@github.com:meltwater/terraform-aws-
modules.git//example?ref=example_v1"

  foo = "bar"
}
```

Meltwater

.drone.yml

pipeline

.drone.yml

```yaml
kind: pipeline
type: docker
name: default

concurrency:
  limit: 1

trigger:
  branch:
  - master

volumes:
- name: dot-aws
  temp: {}
```

.drone.yml

steps

```
- name: files_changed
  image: alpine:3
  commands:
  - apk add git
  - |-
    for DIR in $(git --no-pager diff --name-only
$DRONE_COMMIT_BEFORE..$DRONE_COMMIT_AFTER | sed
"s/^\\(\\.\\/\\)*\\(.*\\)/\\2/" | sed "s/^\\(.*\\)\\/.*/\\1/" |
sort | uniq ); do
      for VERSION in $(find . -name terraform.version | sed
"s/^\\(\\.\\/\\)*\\(.*\\)/\\2/" | sed "s/^\\(.*\\)\\/.*/\\1/" |
sort | uniq); do
        case "$DIR" in
          "$VERSION"*)
            echo $VERSION >> terraform_dirs
            echo "INFO  Adding $VERSION to terrafom_dirs";;
        esac
      done
    done
```

Meltwater

.drone.yml

steps

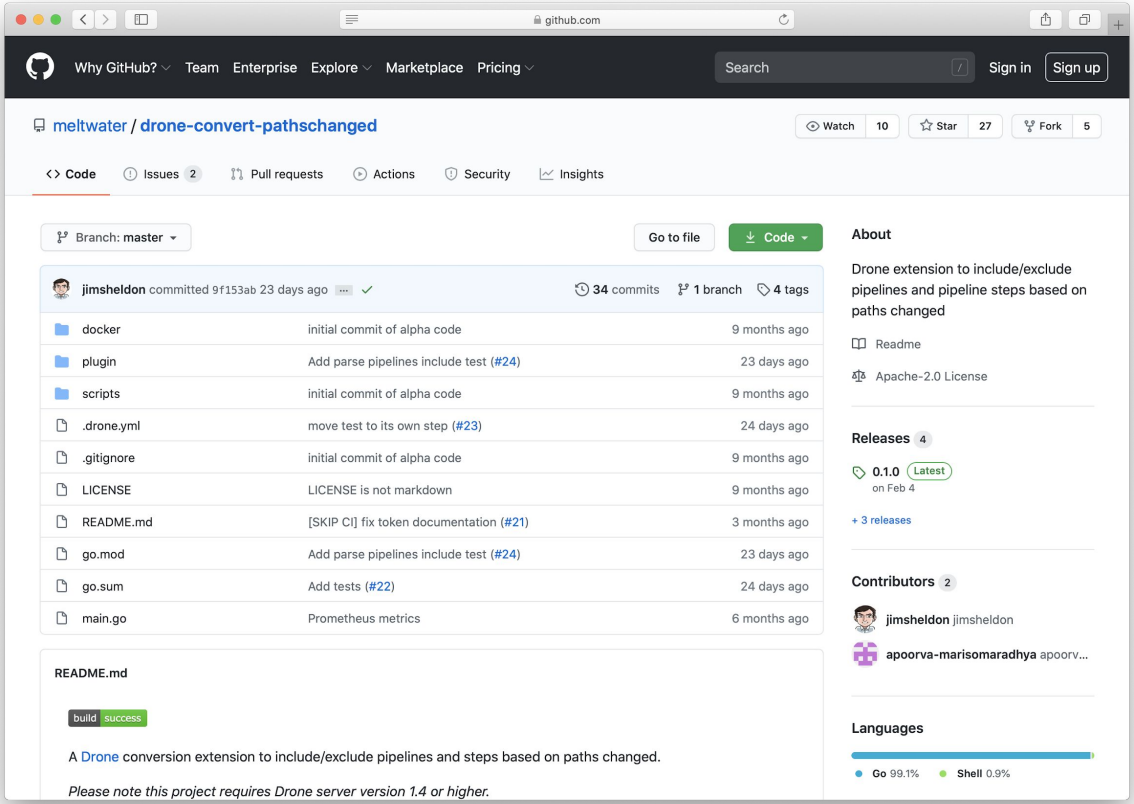
```
- name: aws_credentials
  image: amazon/aws-cli:2.0.31
  volumes:
  - name: dot-aws
    path: /root/.aws
  environment:
    ASSUME_ROLE_EXTERNAL_ID:
      from_secret: assume_role_external_id
    AWS_ACCOUNT_ID:
      from_secret: demo_account_id
  commands:
  - aws configure set role_arn
"arn:aws:iam::$AWS_ACCOUNT_ID:role/DroneTerraformContainerDays"
  - aws configure set region eu-west-1
  - aws configure set credential_source Ec2InstanceMetadata
  - aws configure set external_id $ASSUME_ROLE_EXTERNAL_ID
  - aws sts get-caller-identity
```

.drone.yml

steps

```
- name: terraform_${DRONE_BUILD_EVENT}
  image: alpine:3
  volumes:
  - name: dot-aws
    path: /root/.aws
  environment:
    AWS_METADATA_URL: http://localhost/not/existent/url
    AWS_SDK_LOAD_CONFIG: 1
  commands:
  - apk add bash curl git
  - git clone --branch v1.0.2 https://github.com/tfutils/tfenv.git /opt/tfenv
  - ln -s /opt/tfenv/bin/* /usr/local/bin
  - |-
    ROOT_DIR=$(pwd)
    for DIR in $(cat terraform_dirs); do
      echo "INFO  Changing to $DIR"
      cd $ROOT_DIR/$DIR
      tfenv install $(cat terraform.version)
      tfenv use $(cat terraform.version)
      terraform init
      if [ "$DRONE_BUILD_EVENT" == "pull_request" ]; then
        terraform plan
      elif [ "$DRONE_BUILD_EVENT" == "push" ]; then
        terraform apply
      fi
    done
```

https://github.com/meltwater/drone-convert-pathschanged



Meltwater

conversion

extension

```yaml
kind: pipeline
type: docker
name: default

steps:
- name: readme
  image: busybox
  commands:
  - echo "README.md was changed"
  when:
    paths:
      include:
      - README.md
```

anchors

```yaml
aws_credentials: &aws_credentials
  image: amazon/aws-cli:2.0.31
  volumes:
  - name: dot-aws
    path: /root/.aws
  commands:
  - aws configure --profile $TF_ENV-$TF_REGION set role_arn
"arn:aws:iam::$AWS_ACCOUNT_ID:role/DroneTerraformContainerDays"
  - aws configure --profile $TF_ENV-$TF_REGION set region $TF_REGION
  - aws configure --profile $TF_ENV-$TF_REGION set credential_source Ec2InstanceMetadata
  - aws configure --profile $TF_ENV-$TF_REGION set external_id $ASSUME_ROLE_EXTERNAL_ID


- <<: *aws_credentials
  name: dev_eu-west-1_aws_credentials
  environment:
    ASSUME_ROLE_EXTERNAL_ID:
      from_secret: assume_role_external_id
    AWS_ACCOUNT_ID:
      from_secret: demo_account_id
    TF_ENV: dev
    TF_REGION: eu-west-1
  when:
    paths:
      include:
      - dev/eu-west-1/**
```

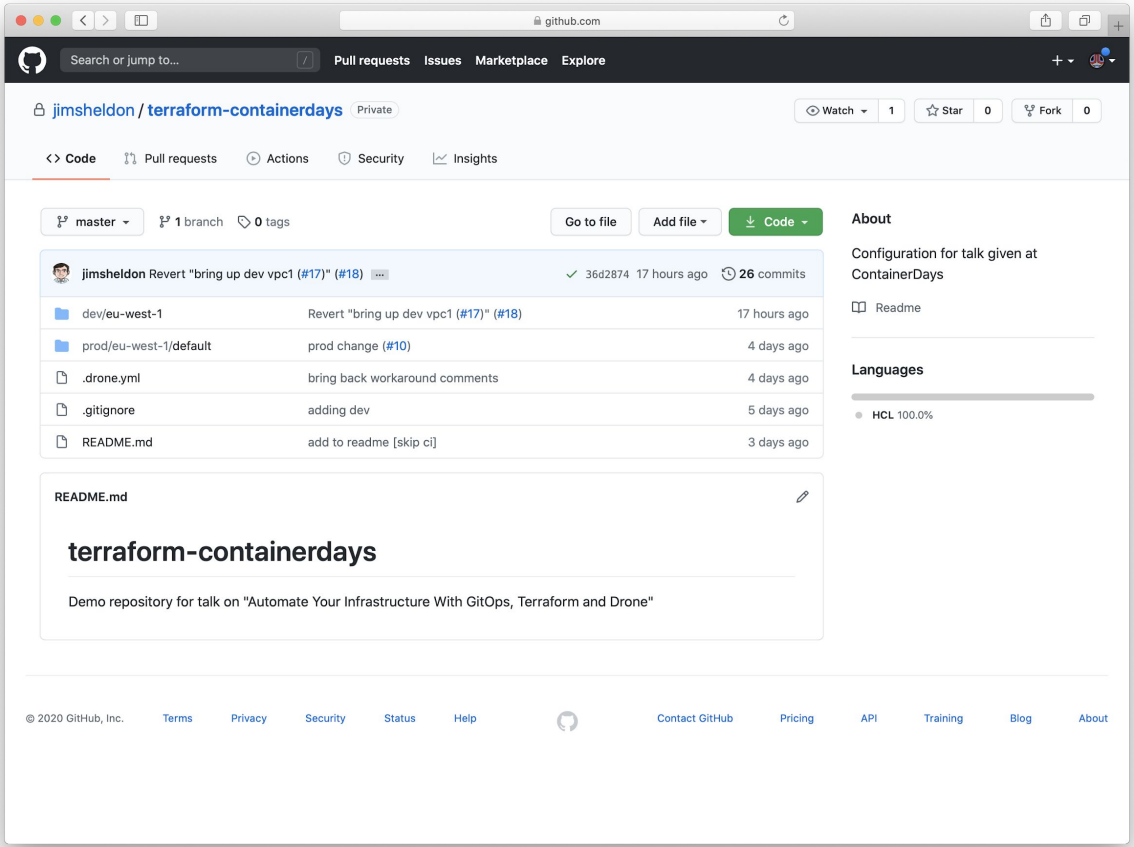
anchors

```yaml
terraform: &terraform
  image: alpine:3
  volumes:
  - name: dot-aws
    path: /root/.aws
  commands:
  - apk add bash curl git
  - git clone --branch v1.0.2 https://github.com/tfutils/tfenv.git /opt/tfenv
  - ln -s /opt/tfenv/bin/* /usr/local/bin
  - |-
    ROOT_DIR=$(pwd)
    for DIR in $(cat terraform_dirs | grep ^$TF_ENV/$TF_REGION); do
      echo "INFO  Changing to $DIR"
      cd $ROOT_DIR/$DIR
      tfenv install $(cat terraform.version)
      tfenv use $(cat terraform.version)
      terraform init
      if [ "$DRONE_BUILD_EVENT" == "pull_request" ]; then
        terraform plan
      elif [ "$DRONE_BUILD_EVENT" == "push" ]; then
        terraform apply
      fi
    done

- <<: *terraform
  name: dev_eu-west-1_terraform_${DRONE_BUILD_EVENT}
  environment:
    AWS_METADATA_URL: http://localhost/not/existent/url
    AWS_SDK_LOAD_CONFIG: 1
    AWS_PROFILE: dev-eu-west-1
    TF_ENV: dev
    TF_REGION: eu-west-1
  when:
    paths:
      include:
      - dev/eu-west-1/**
```

Meltwater

DEMO!

https://github.com/jimsheldon/terraform-containerdays



Meltwater

IAM role

(your account)

```
resource "aws_iam_role" "role_drone" {
  name               = "DroneTerraformContainerDays"
  assume_role_policy = data.aws_iam_policy_document.assume_policy_drone.json
}

data "aws_iam_policy_document" "assume_policy_drone" {
  statement {
    effect = "Allow"
    principals {
      identifiers = [
        data.aws_kms_secrets.drone.plaintext["trusted_account_arn"],
      ]
      type = "AWS"
    }
    actions = ["sts:AssumeRole"]
    condition {
      test     = "StringEquals"
      values   = [data.aws_kms_secrets.drone.plaintext["external_id"]]
      variable = "sts:ExternalId"
    }
  }
}
```

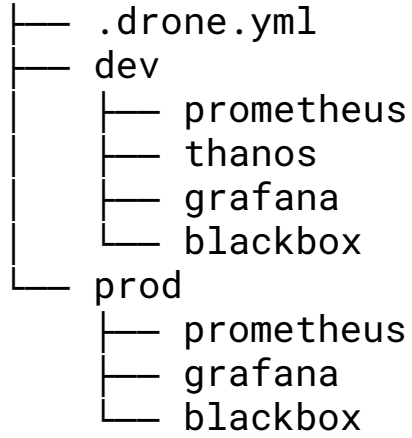
IAM role

(drone account)

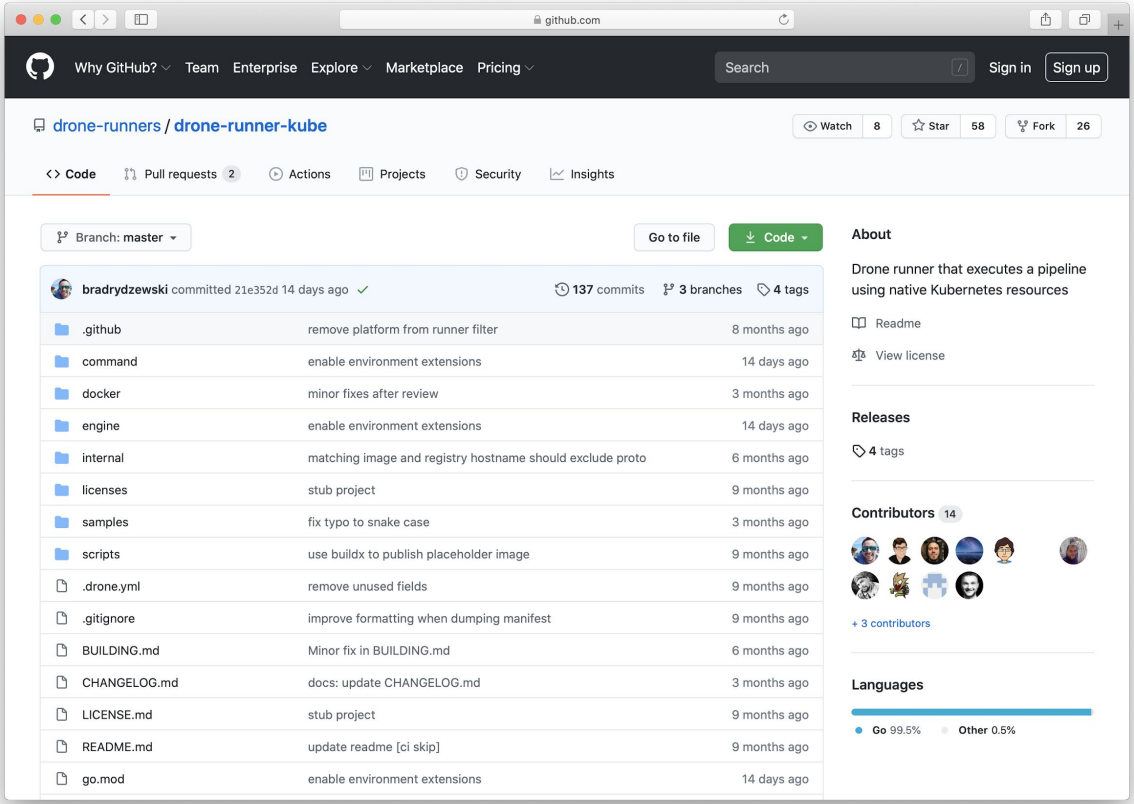
```
data "aws_iam_policy_document" "foundation" {
  "statement" {
    effect    = "Allow"
    actions   = ["sts:AssumeRole"]
    resources = [
      "arn:aws:iam::${var.foundation_account_id}:role/DroneTerraformContainerDays",
    ]
  }
}

resource "aws_iam_role_policy" "foundation" {
  name   = "allow_foundation_assume"
  role   = "${data.aws_iam_role.drone_agent.id}"
  policy = "${data.aws_iam_policy_document.foundation.json}"
}
```

kubernetes

K8S directory structure



https://github.com/drone-runners/drone-runner-kube

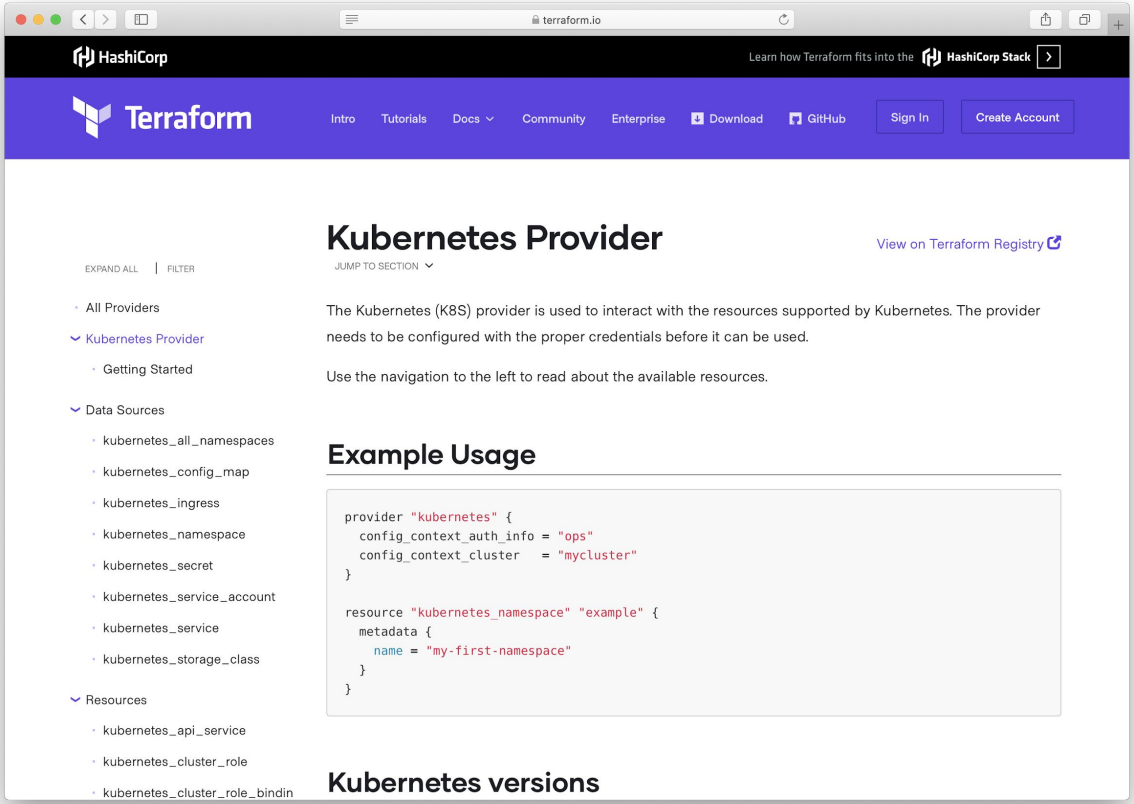


.drone.yml

k8s runner

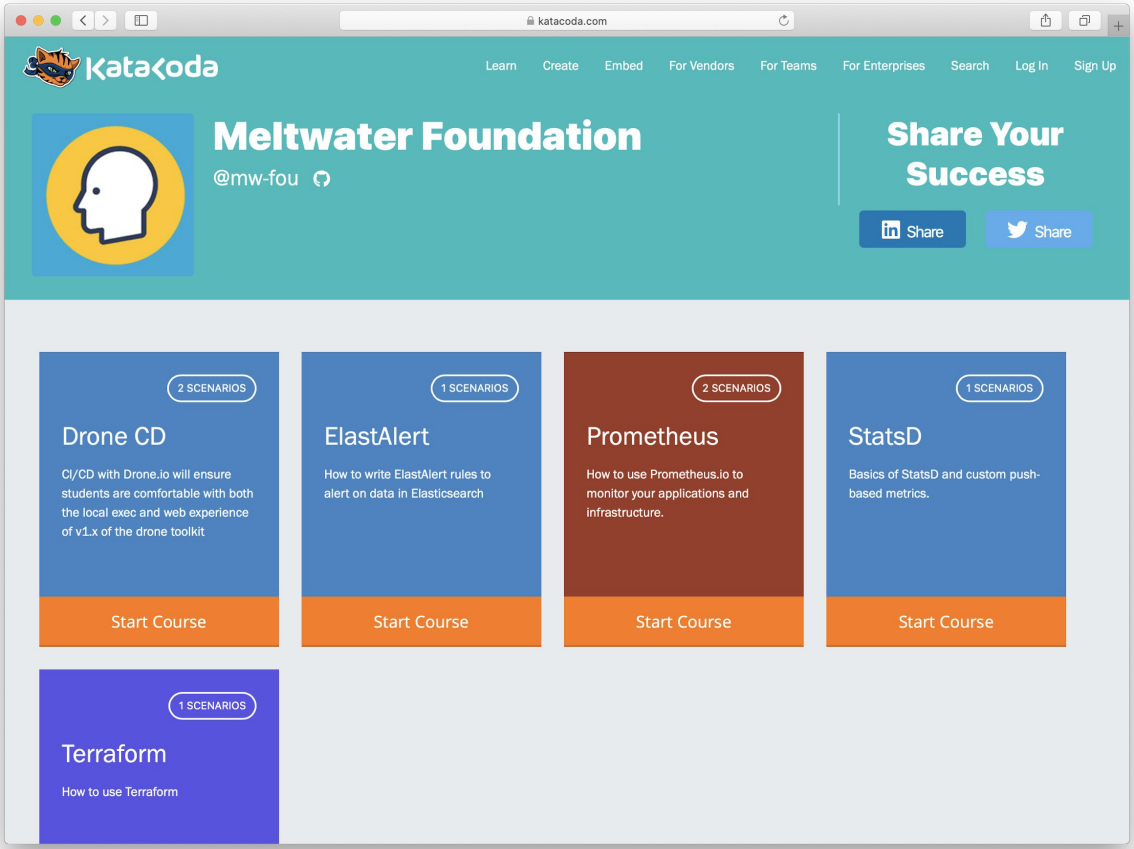
```
kind: pipeline
type: kubernetes
name: default

steps:
- name: greeting
  image: alpine
  commands:
  - echo "hello world"
```

https://www.terraform.io/docs/providers/kubernetes/index.html



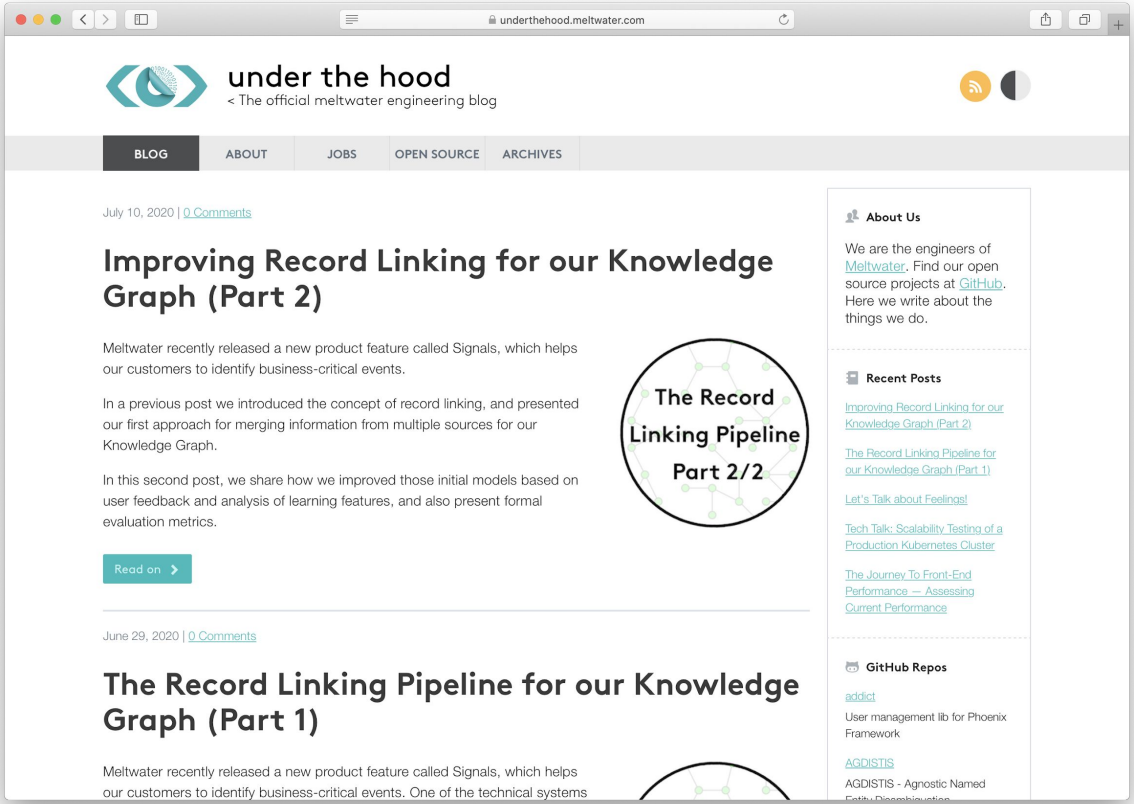
Meltwater

https://katacoda.com/mw-fou



Meltwater

https://underthehood.meltwater.com



Meltwater

Thank you!

Questions?