Pipelines as Code for Infrastructure at Scale

Pipelines as Code

Pipelines as Code for Infrastructure

Pipelines as Code for Infrastructure at Scale

Agenda

A problem to solve

An attempt

The Issue

What are the options?

Programmatic pipelines

Scaling up

Results

Takeaways

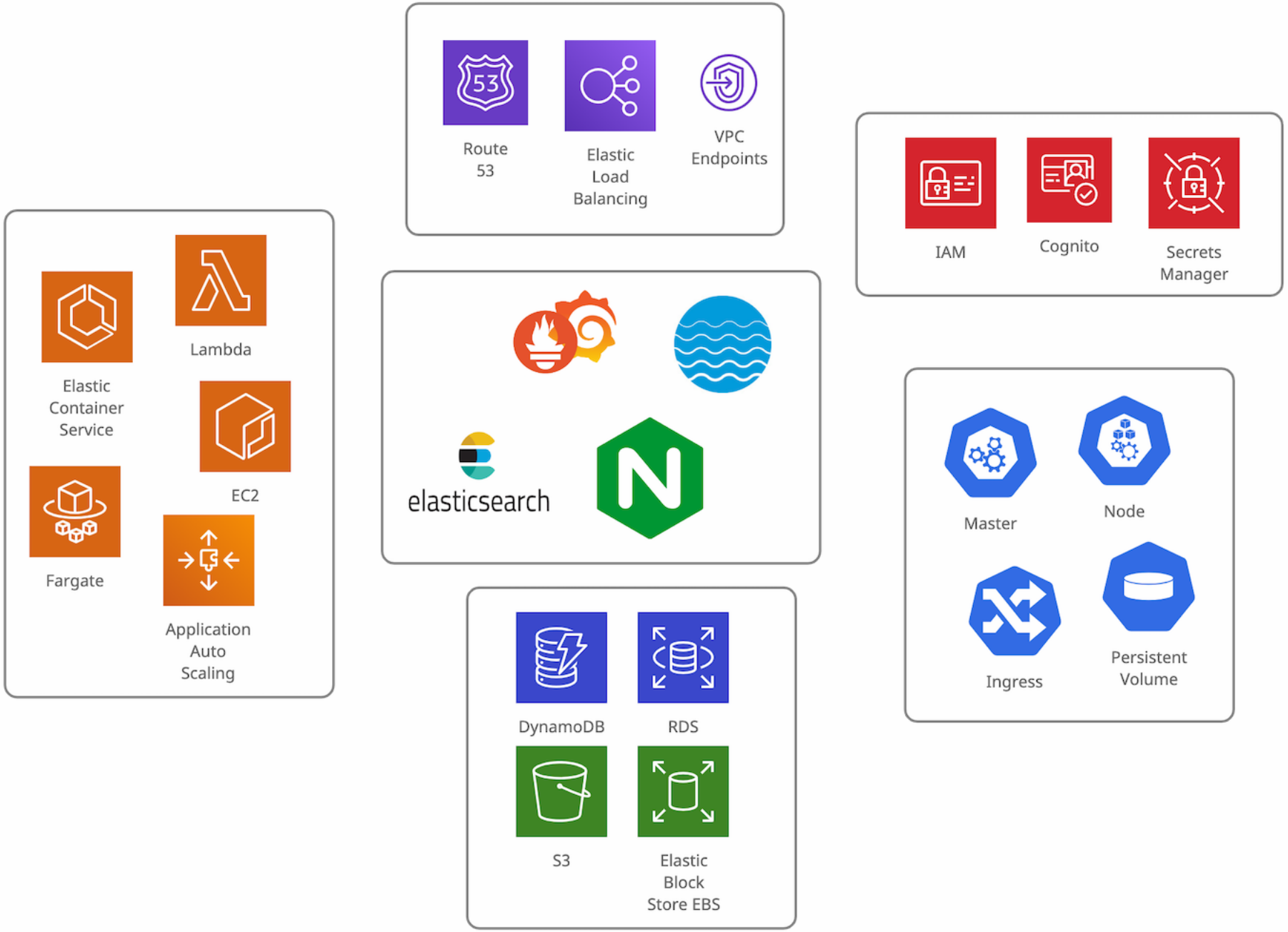
Mario Fernandez

ThoughtWorks

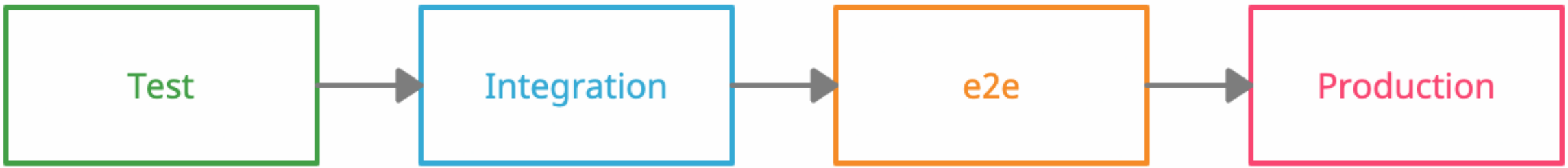
A problem to solve

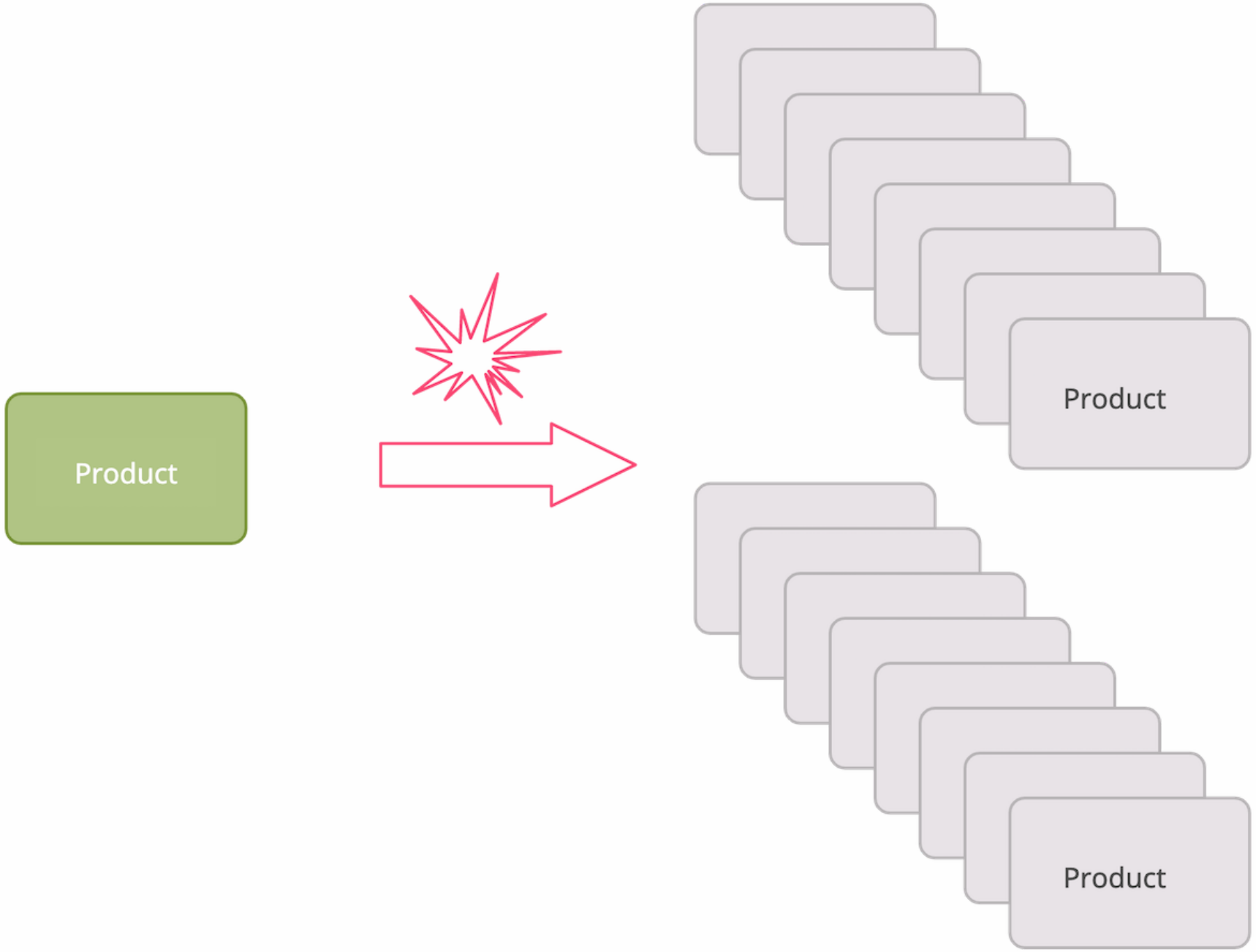
Enable teams to move from on-premise to the cloud

Layer on top of AWS to provide a ready-to-use solution


Route 53
Elastic Load Balancing
VPC Endpoints
IAM
Cognito
Secrets Manager
Lambda
Elastic Container Service
EC2
Fargate
Application Auto Scaling
elasticsearch
Master
Node
Ingress
Persistent Volume
DynamoDB
RDS
S3
Elastic Block Store EBS


© 2009 www.outline-world-map.com


Test
Integration
e2e
Production


Product
Product
Product

An attempt

Provisioning a ton of infrastructure

```
modules
├── access-from-other-accounts
├── application-environment-1
├── application-environment-2
├── cluster-{backup,management}
├── connected-hosted-zone
├── dispatcher-vpc
├── domain
├── functional-area
├── functional-area-{access,runtime}
├── logging
├── monitoring
├── operations
├── private-egress
├── private-ingress-{app,dispatcher}
└── system-services
```

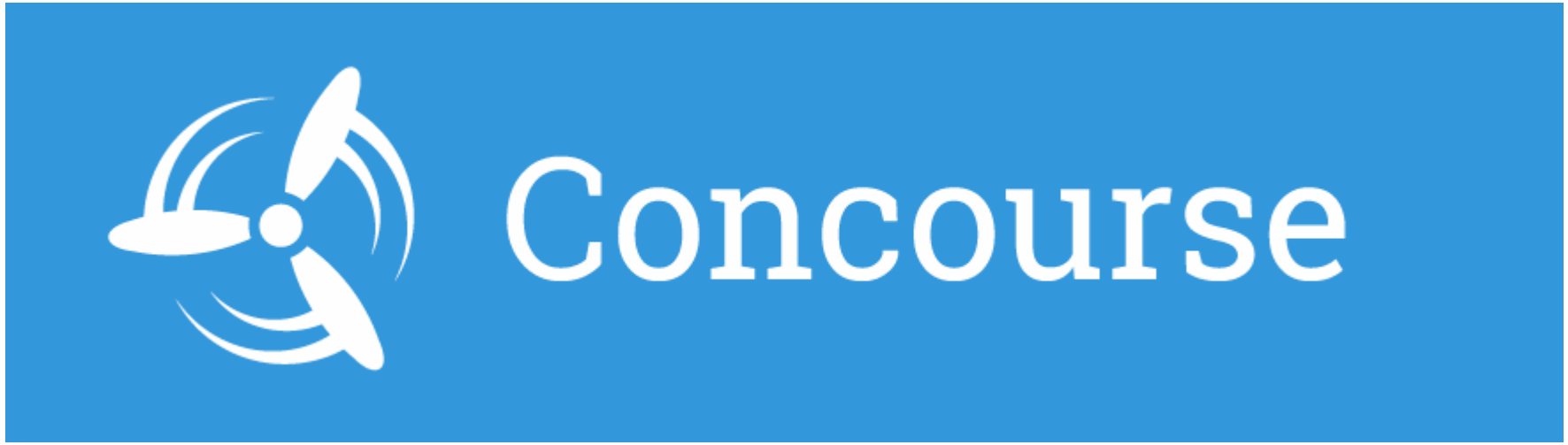
```
modules/
├── access-from-other-accounts
|   ├── dependencies.tf
|   ├── locals.tf
|   ├── main.tf
|   ├── outputs.tf
|   ├── providers.tf
|   └── variables.tf
├── application-environment-1
|   ├── data.tf
|   ├── locals.tf
|   ├── main.tf
|   ├── outputs.tf
|   ├── providers.tf
|   └── variables.tf
└── tracing
```

```
[terragrunt] 2021/02/01 20:28:39 Executing hook: tflint
[terragrunt] 2021/02/01 20:28:39 Running command: tflint
[terragrunt] 2021/02/01 20:28:40 Running command: terraform apply
module.aws_vpc.module.vpc.aws_subnet.private[2]: Refreshing state..
module.aws_vpc.module.vpc.aws_vpc.this[0]: Refreshing state..
module.aws_vpc.module.vpc.aws_subnet.private[0]: Refreshing state..
module.aws_vpc.module.vpc.aws_subnet.private[1]: Refreshing state..
module.aws_vpc.module.vpc.aws_vpc.this[0]: Refreshing state..
module.aws_eks.data.aws_region.current: Refreshing state..

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
```

Spoiler Alert

Doing it by hand did not scale



thoughtworks.com/radar/techniques/pipelines-for-infrastructure-as-code

Apply infrastructure changes automatically

Small modules

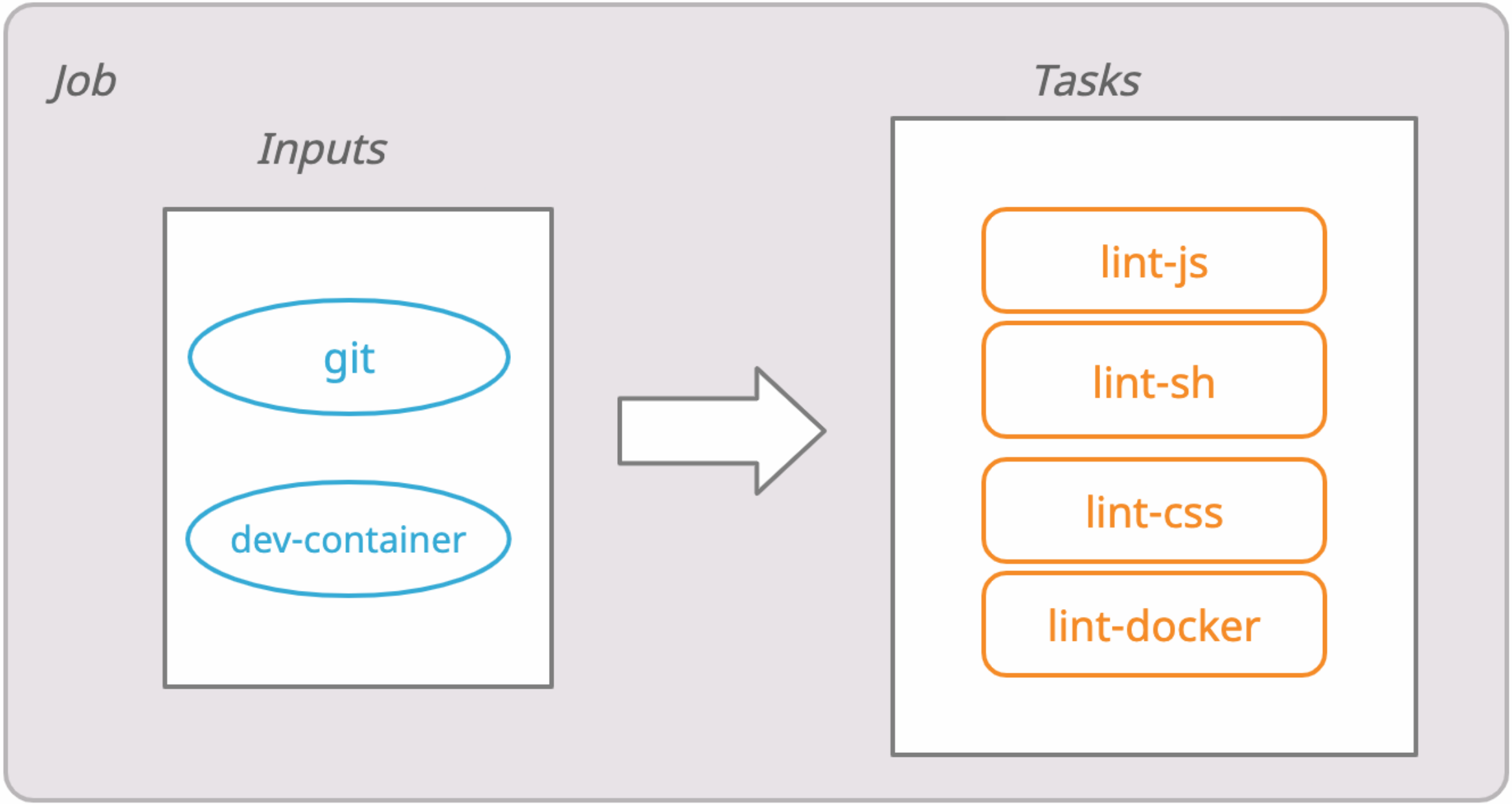
Pipeline is an orchestrator of existing scripts

The issue

YAML is really verbose

Like, really

Concourse doesn't help, either


Job
Inputs
git
dev-container
Tasks
lint-js
lint-sh
lint-css
lint-docker

```yaml
- name: lint
  serial: true
  plan:
  - in_parallel:
    - get: git
      passed: [prepare]
      trigger: true
    - get: dev-container
      passed: [prepare]
  - in_parallel:
    - task: lint-sh
      image: dev-container
      params:
        <<: *common-params
        TARGET: sh
      file: git/pipeline/tasks/linter/task.yml
```

Our ability to manage duplication is limited

Anchor parameters

```yaml
  - in_parallel:
    - task: lint-sh
      image: dev-container
      params:
        <<: *common-params
        TARGET: sh
      file: git/pipeline/tasks/linter/task.yml
```

```yaml
common-params: &common-params
  CI: true
```

Parametrized tasks

```
  - in_parallel:
    - task: lint-sh
      image: dev-container
      params:
        <<: *common-params
        TARGET: sh
      file: git/pipeline/tasks/linter/task.yml
```

YAML overdose!

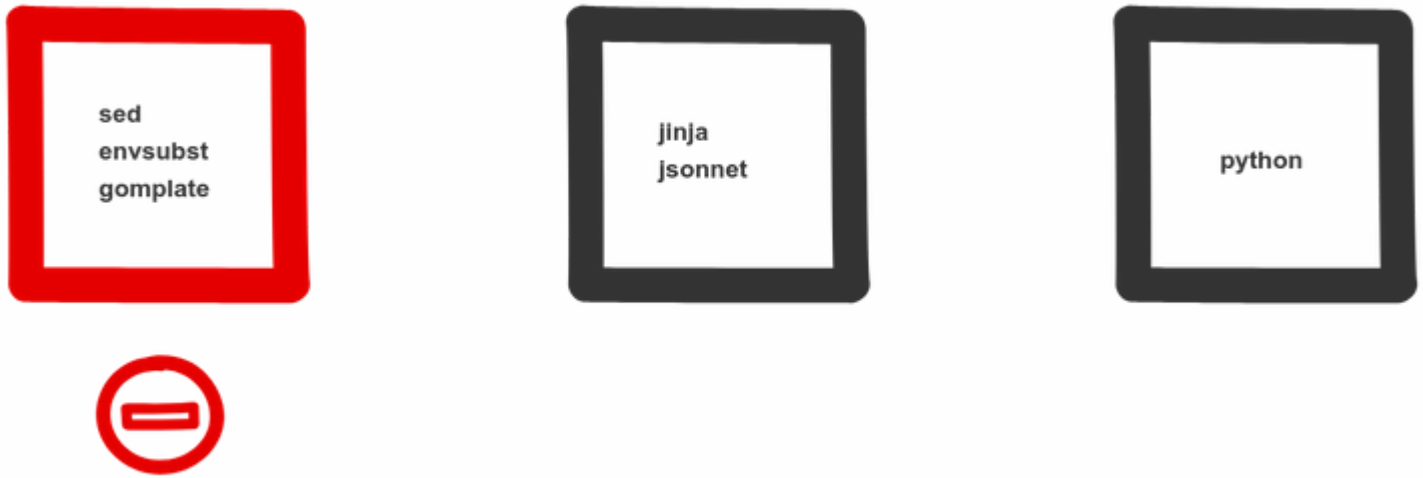
```yaml
  serial: true
- name: cluster-base-network-layout-test
  plan:
  - in_parallel:
    - get: source-code
      passed:
      - cluster-global-dns
      - functional-area-account-dev
      trigger: true
    - get: dev-tools
      trigger: false
    - get: terraform-tools
      trigger: false
    - get: container
      trigger: false
  - in_parallel:
    - file: source-code/ci/tasks/provision.yaml
      image: container
      params:
        AUTOMATION: true
        CI_AWS_ACCESS_KEY_ID: ((aws-access-key-id))
        CI_AWS_SECRET_ACCESS_KEY: ((aws-secret-access-key))
        CONCOURSE_ADMIN_PASSWORD: ((admin-password))
        CONCOURSE_ADMIN_USERNAME: ((admin-username))
        REGION: eu-central-1
      task: domain-eu-central-1
  - in_parallel:
    - file: source-code/ci/tasks/provision.yaml
      image: container
      params:
        AUTOMATION: true
        CI_AWS_ACCESS_KEY_ID: ((aws-access-key-id))
        CI_AWS_SECRET_ACCESS_KEY: ((aws-secret-access-key))
```

What are the options?

Something on top of YAML

Three approaches


sed
envsubst
gomplate
jinja
jsonnet
python


sed
envsubst
gomplate
jinja
jsonnet
python

Templating in YAML

APR 2019

HOLD

As infrastructures grow in complexity, so do the configuration files that define them. Tools such as **AWS CloudFormation**, **Kubernetes** and **Helm** expect configuration files in JSON or YAML syntax, presumably in an attempt to make them easy to write and process. However, in most cases, teams quickly reach the point where they have some parts that are similar but not quite the same, for example, when the same service must be deployed in different regions with a slightly different setup. For such cases tools offer **templating in YAML** (or JSON), which has caused a huge amount of **frustration with practitioners**. The problem is that the syntax of JSON and YAML requires all sorts of awkward compromises to graft templating features such as conditionals and loops into the files. We recommend using an API from a programming language instead or, when this is not an option, a templating system in a programming language, either a general-purpose language such as Python or something specialized such as **Jsonnet**.

> We recommend using an API from a programming language, or, a templating system

Programmatic pipelines

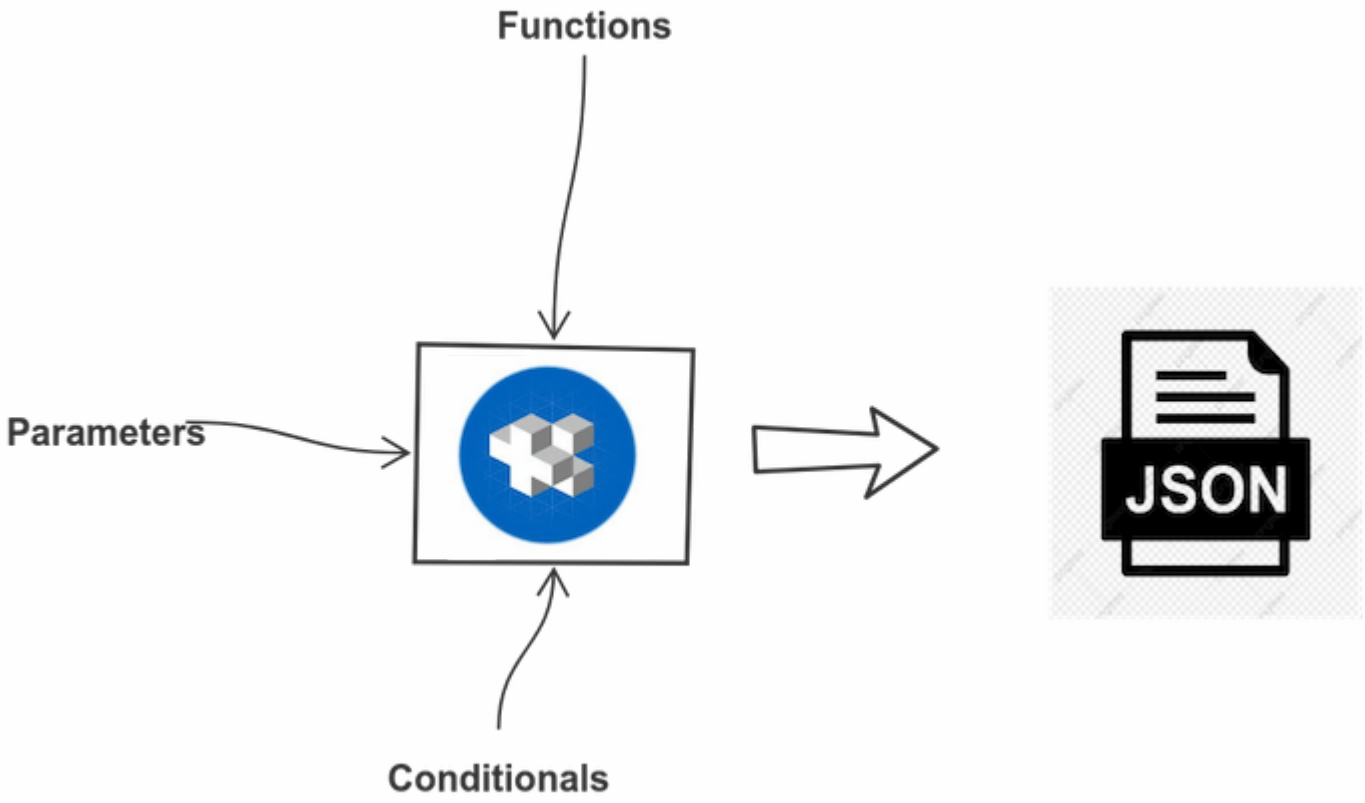
Generate most of the pipeline automatically

Handle duplication through a parametrized approach

Jsonnet

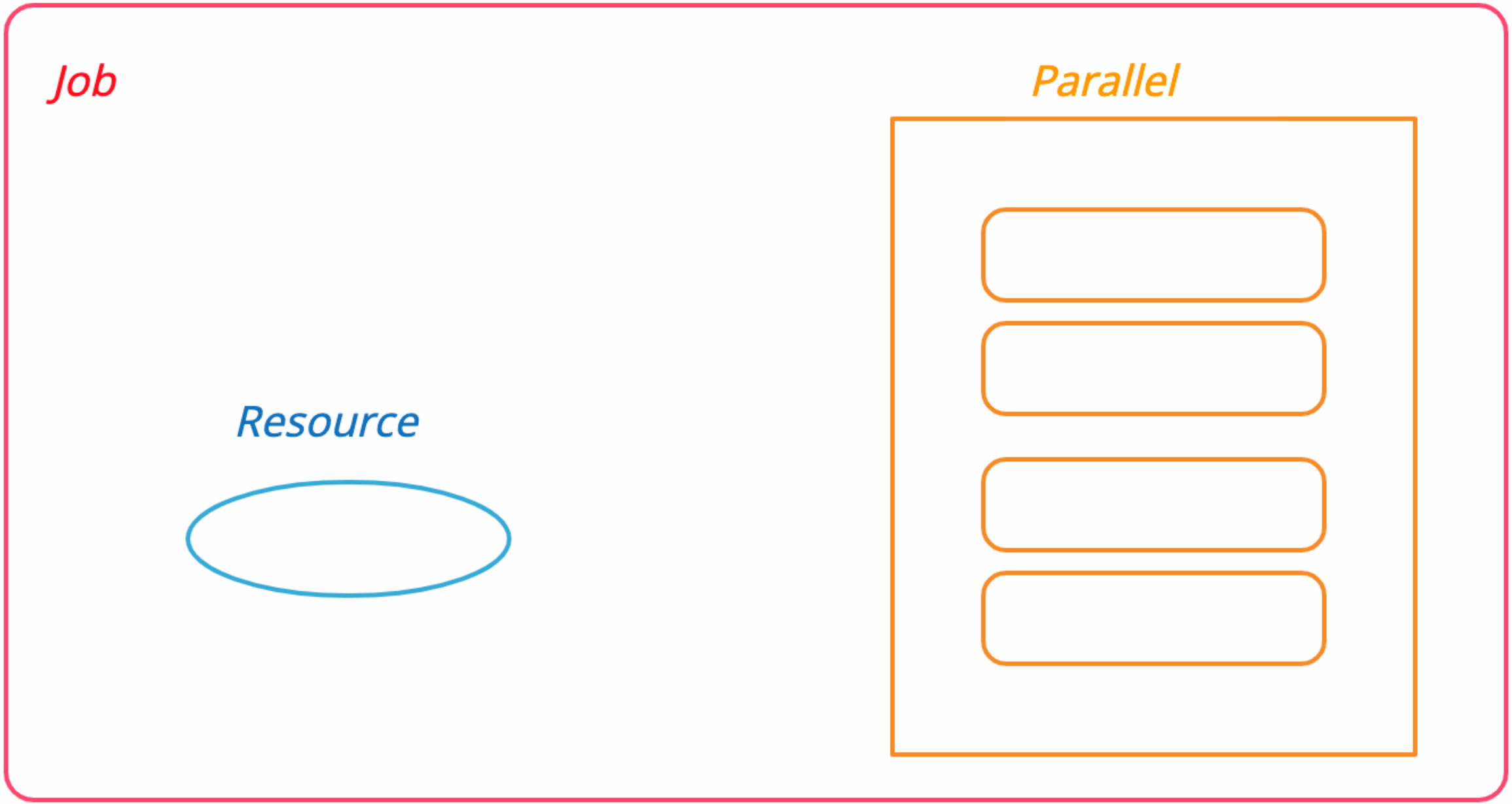
jsonnet.org/

A data templating language for app and tool developers


Functions
Parameters
Conditionals
JSON

YAML happens to be a superset of JSON

Abstract building blocks


Job
Parallel
Resource

```
local Parallel(tasks) = {
  in_parallel: tasks
}
```

```
local Job(name, serial = true, plan = []) = std.prune({
  name: name,
  serial: serial,
  plan: plan
})
```

```
DockerResource(name,
               repository,
               tag = 'latest', allow_insecure = false) = {
  name: name,
  type: 'docker-image',
  source: {
    repository: repository,
    tag: tag
  } + (
    if allow_insecure then {
      insecure_registries: [std.split(repository, '/')[0]]} else {}
  ),
}
```

```
local concourse = import 'concourse.libsonnet';
{
  docker: concourse.DockerResource('serverspec-container',
                                   'sirech/dind-ruby', tag = '2.6.3'),
}
```

```
{
   "docker": {
      "name": "serverspec-container",
      "source": {
         "repository": "sirech/dind-ruby",
         "tag": "2.6.3"
      },
      "type": "docker-image"
   }
}
```

github.com/sirech/concourse-jsonnet-utils

Building your own DSL

Reflect your conventions and idioms in code

```
local source = 'git';
local container = 'dev-container';

local concourse = import 'concourse.libsonnet';

local Inputs(dependencies = []) = concourse.Parallel(
  [concourse.Get(s, dependencies = dependencies)
    for s in [source, container]]
);
```

```
local Task(name, file = name, image = container, params = {}) = {
  task: name,
  image: image,
  params: {
    CI_AWS_ACCESS_KEY_ID: ((aws-access-key-id)),
    CI_AWS_SECRET_ACCESS_KEY: ((aws-secret-access-key)),
    MODULE: module,
    PRODUCT: product,
    CI: true
  } + params,
  file: '%s/pipeline/tasks/%s/task.yml' % [source, file]
};
```

Back to the original example

```yaml
- name: lint
  serial: true
  plan:
  - in_parallel:
    - get: git
      passed: [prepare]
      trigger: true
    - get: dev-container
      passed: [prepare]
  - in_parallel:
    - task: lint-sh
      image: dev-container
      params:
        <<: *common-params
        TARGET: sh
      file: git/pipeline/tasks/linter/task.yml
```

```
local concourse = import 'concourse.libsonnet';

concourse.Job('lint', plan = [
  Inputs('prepare'),
  concourse.Parallel(
    [Task('lint-%s' % lang, 'linter', params = { TARGET: lang })
      for lang in ['sh', 'js', 'css', 'docker']]
  )
]),
```

github.com/sirech/example-concourse-pipeline

```
local product = std.extVar('SERVICE_NAME');
local config = std.extVar('CONFIG');

local concourse = import 'concourse.libsonnet';
local envs = import 'environments.libsonnet';
local builders = import 'builders.libsonnet';

local TaskPreparePipeline(name, file, image='prepare-pipeline-container') = builders.Task(name=name, file=file, image=image);
local TaskPerRegion(module, regions, params={}) = concourse.Parallel([builders.Task('%s-%s' % [module, region], module, file=if std.startsWith(module, 'smoketest') then 'test' else 'provision', params=params { REGION: region }) for region in regions]);
local TaskPerArea(module, params={}) = concourse.Parallel([builders.Task('%s-%s' % [module, area], module, params=params { AREA: area }) for area in envs.areas]);
local TaskPerAreaAndRegion(module, regions, params={}) = concourse.Parallel([builders.Task('%s-%s-%s' % [module, area, region], module, params=params { AREA: area, REGION: region }) for area in envs.areas for region in regions]);
local TestPerRegion(module, regions, params={}) = concourse.Parallel([builders.Task('%s-%s' % [module, region], module, file='intranet-conn-test', params=params { REGION: region }) for region in regions]);

local Job(name, dependencies=[], trigger=true, tasks=[], sources=['dev-tools', 'terraform-tools', 'container', 'accounts']) =
  local toA(t) = if std.isArray(t) then t else [t];
  local plan = [concourse.Parallel(builders.Inputs(toA(dependencies), trigger=trigger, sources=sources))] + toA(tasks);
  concourse.Job(name, plan=plan)
;

local resources = [
  builders.GitResource('source-code', 'products', paths=['modules', product]),
  builders.GitResource('dev-tools', 'dev-tools'),
  builders.GitResource('accounts', 'accounts'),
  builders.GitResource('terraform-tools', 'terraform-tools'),
  builders.DockerResource('container', '${ACCOUNT}.dkr.ecr.eu-central-1.amazonaws.com/$project/public/util-terraform', '0.12'),
  builders.DockerResource('prepare-pipeline-container', '${ACCOUNT}.dkr.ecr.eu-central-1.amazonaws.com/$project/public/util-pipeline', '0.1'),
];


local globalJobs = [
  Job('prepare-pipeline', tasks=[TaskPreparePipeline('validate-product-yaml', 'product-yaml-validation'), TaskPreparePipeline('generate-pipeline', 'pipeline-generation')], sources=['prepare-pipeline-container']),
  Job('functional-area-global',dependencies=['prepare-pipeline'] ,tasks=TaskPerArea('functional-area')),
  Job('keycloak', dependencies=['prepare-pipeline'], tasks=builders.Task('keycloak'))
];

local accounts = [
  { account: 'dev', dependencies: [job.name for job in globalJobs] },
  { account: 'prod', dependencies: ['smoketest-%s' % [env] for env in envs.nonProdEnvironments] },
];

local accountFunctionalArea = [
  Job('functional-area-account-%s' % [account.account], account.dependencies, trigger=envs.continuousDeployment(account.account), tasks=TaskPerArea('functional-area-access', { ACCOUNT: account.account }))
  for account in accounts
];

local accountNetwork = [
  Job(
    'dispatcher-vpc-%s' % [account.account],
    'functional-area-account-%s' % [account.account],
    tasks=[builders.Task('dispatcher-vpc', 'dispatcher-vpc', params={ ACCOUNT: account.account }),builders.Task('acme-dispatcher-endpoint', 'acme-dispatcher-endpoint', params={ ACCOUNT: account.account })]
  )
  for account in accounts
];

local accountJobs = accountFunctionalArea + accountNetwork;

local EnvironmentJobs(env) =
  local regions = std.objectFields(config.clusters[env]);
  local params = { ENV: env };

  local modules = [
    Job('cluster-base-network-layout-%s' % [env], ['cluster-global-dns', 'functional-area-account-%s' % envs.accountFromEnv(env)] + envs.previousSmoketest(env), tasks=[TaskPerRegion(module, regions, params) for module in ['domain', 'application-environment-1', 'application-environment-2', 'operations']]),
    Job('customer-resources-vpc-%s' % [env], ['cluster-base-network-layout-%s' % [env]], tasks=[TaskPerRegion(module, regions, params) for module in ['customer-resources-vpc']]),
    Job('cluster-services-%s' % [env], 'cluster-base-network-layout-%s' % [env], tasks=[TaskPerRegion(module, regions, params) for module in ['system-services', 'cluster-backup', 'cluster-management', 'monitoring', 'logging', 'tracing']]),
    Job('functional-area-runtime-%s' % [env], 'cluster-services-%s' % [env], tasks=TaskPerAreaAndRegion('functional-area-runtime', regions, params)),
    Job('ingress-%s' % [env], ['cluster-services-%s' % [env], 'dispatcher-vpc-%s' % envs.accountFromEnv(env)], tasks=[TaskPerRegion(module, regions, params) for module in ['private-ingress-app', 'private-ingress-dispatcher', 'access-from-other-accounts']]),
    Job('egress-%s' % [env], ['cluster-base-network-layout-%s' % [env], 'ingress-%s' % [env]], tasks=[
      TaskPerRegion('private-egress', regions, params),
      TaskPerRegion('private-egress-mq-server', regions, params),
      TestPerRegion('intranet-conn-test', regions, (params + {ACCOUNT: envs.accountFromEnv(env)})),
      TaskPerRegion('private-egress-acme', regions, (params + {ACCOUNT: envs.accountFromEnv(env)}))
    ]),
    Job('hello-world-%s' % [env], 'cluster-services-%s' % [env], tasks=TaskPerRegion('hello-world', regions, params)),
  ];

  modules + [
    Job('smoketest-%s' % [env], ['%s' % [module] for module in [job.name for job in modules]], tasks=[
      TaskPerRegion('smoketest', regions, params),
    ]),
  ]
;
local nonProdEnvironmentJobs = [{ env: env, jobs: EnvironmentJobs(env) } for env in config.environment_order if std.member(envs.nonProdEnvironments, env)];
local prodEnvironmentJobs = [{ env: env, jobs: EnvironmentJobs(env) } for env in config.environment_order if std.member(envs.prodEnvironments, env)];

local scopedGroups = std.flattenArrays([
  [concourse.Group('global', [job.name for job in globalJobs])],
  [concourse.Group('account-dev', [job.name for job in accountJobs if std.endsWith(job.name, 'dev')])],
  [concourse.Group('account-prod', [job.name for job in accountJobs if std.endsWith(job.name, 'prod')])],
  [concourse.Group(jobGroup.env, [job.name for job in jobGroup.jobs]) for jobGroup in nonProdEnvironmentJobs],
  [concourse.Group(jobGroup.env, [job.name for job in jobGroup.jobs]) for jobGroup in prodEnvironmentJobs],
]);

local groups = [{
  name: 'all',
  jobs: std.flatMap(function(x) x.jobs, scopedGroups),
}] + scopedGroups;

{
  groups: groups,
  resources: resources,
  jobs: globalJobs + accountJobs + std.flattenArrays([environment.jobs for environment in nonProdEnvironmentJobs + prodEnvironmentJobs]),
}
```

Scaling up

Next goal

Generate many pipelines

```
product_name: new-product
version: master

clusters:
  test:
    eu-west-1:
  prod:
    eu-west-1:
    us-east-1:

nodes:
  test:
    instance_type: "m5.large"
  prod:
    instance_type: "c5.2xlarge"
```

```
local config = std.extVar('CONFIG')
```

```
# Convert the config to JSON and merge it, so that values are overriden
CONFIG="$(yq -r '. * .' ../product_defaults.yaml product.yaml \
          | jq -s 'add')"
```

```
local EnvironmentJobs(env) =
  local regions = std.objectFields(config.clusters[env]);
  local params = { ENV: env };
  local services = ['system-services', 'monitoring', 'logging'];

  local modules = [
    Job('cluster-services-%s' % [env],
        'cluster-global-dns',
        tasks=[TaskPerRegion(module, regions, params)
               for module in services]),
  ];

  modules + [
    Job('smoketest-%s' % [env],
        ['%s' % [module] for module in [job.name for job in modules]],
        tasks=[TaskPerRegion('smoketest', regions, params)]),
  ]
```

Only works up to a certain point!

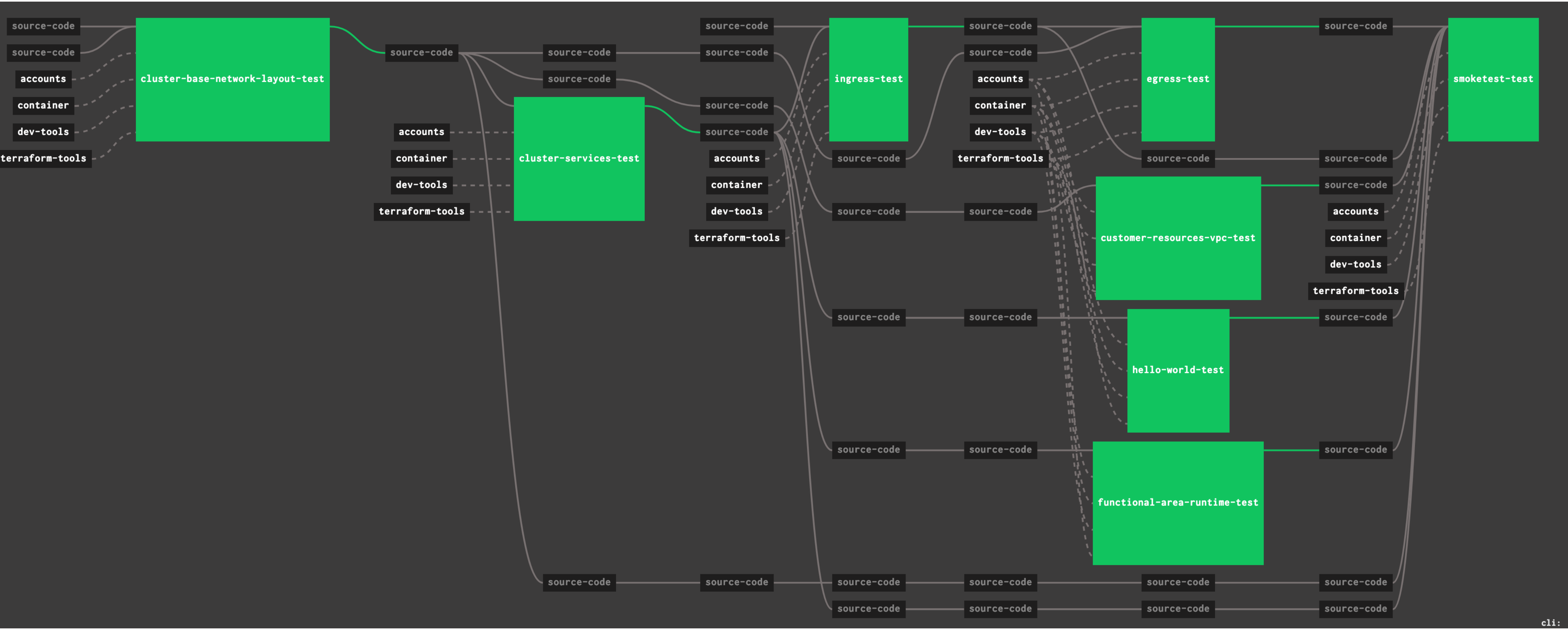
Some results

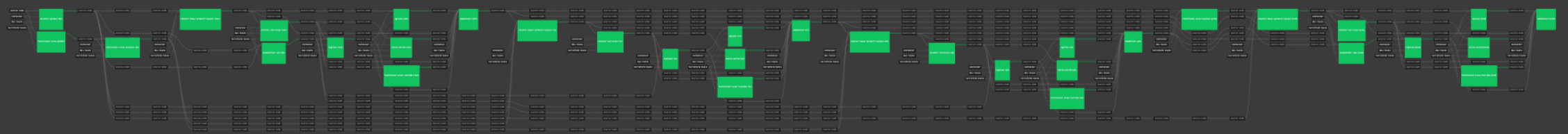
cluster-services-prod #204

started 5h 32m ago
finished 5h 18m ago
duration 14m 18s

204 203 202 201 200 199 198 197 196 195 194 193 192 191 190 189 188 187 186 185 184 183 182 181 180 179 178 177 176 175 174 173 172 171 170 169 168 167 166 165 164 163 162 161 160 159 158 157 156 155 154

source-code	ref	1ac205f3ebc1e06d74f529c952e65442932d784a
dev-tools	ref	fce0859c9663b316fe60f5251623988ab4ef3f24
terraform-tools	ref	554b9093eaa01e1a01f5986b558d48479a2d63ee
container	digest	sha256:f25a3a7bf3e865ec290b952badebbccddde8ac3475d518f7783835bb969e5d78
accounts	ref	5832e089d5ec31784d8f61039f6abf29603be533
system-services-ap-northeast-2		
system-services-eu-central-1		
system-services-us-east-1		
cluster-backup-ap-northeast-2		
cluster-backup-eu-central-1		
cluster-backup-us-east-1		
cluster-management-ap-northeast-2		
cluster-management-eu-central-1		
cluster-management-us-east-1		
monitoring-ap-northeast-2		
monitoring-eu-central-1		
monitoring-us-east-1		
logging-ap-northeast-2		
logging-eu-central-1		
logging-us-east-1		
tracing-ap-northeast-2		
tracing-eu-central-1		
tracing-us-east-1		


source-code
source-code
accounts
container
dev-tools
terraform-tools
cluster-base-network-layout-test
source-code
accounts
container
dev-tools
terraform-tools
source-code
source-code
cluster-services-test
source-code
source-code
source-code
source-code
source-code
accounts
container
dev-tools
terraform-tools
source-code
ingress-test
source-code
source-code
source-code
source-code
source-code
source-code
source-code
source-code
accounts
container
dev-tools
terraform-tools
source-code
source-code
source-code
source-code
source-code
egress-test
source-code
customer-resources-vpc-test
hello-world-test
functional-area-runtime-test
source-code
source-code
source-code
source-code
source-code
accounts
container
dev-tools
terraform-tools
source-code
source-code
source-code
source-code
smoketest-test
cli:



space-lab

running

space-lab-cn

13d 4h

product1

19h 57m

product1-cn

51m 13s

general-services

20h 3m

special-services

19h 56m

product2

running

product3

20h 13m

product3-cn

20h 7m

adc

19h 59m

bbs

20h 1m

product4

19h 59m

product4-cn

19h 55m

other

18h 14m

sii

19h 59m

sii-cn

19h 33m

Generated	LOC
pipeline.yaml	7312

Jsonnet	LOC
pipeline.jsonnet	102
concourse.libsonnet	54
builders.libsonnet	46
environments.libsonnet	19

hceris.com/templating-concourse-pipelines-with-jsonnet/

Takeaways

Use Infrastructure as Code

Use Infrastructure Pipelines

Invest in your tooling