EESSI : BEHIND THE SCENES

INFRASTRUCTURE

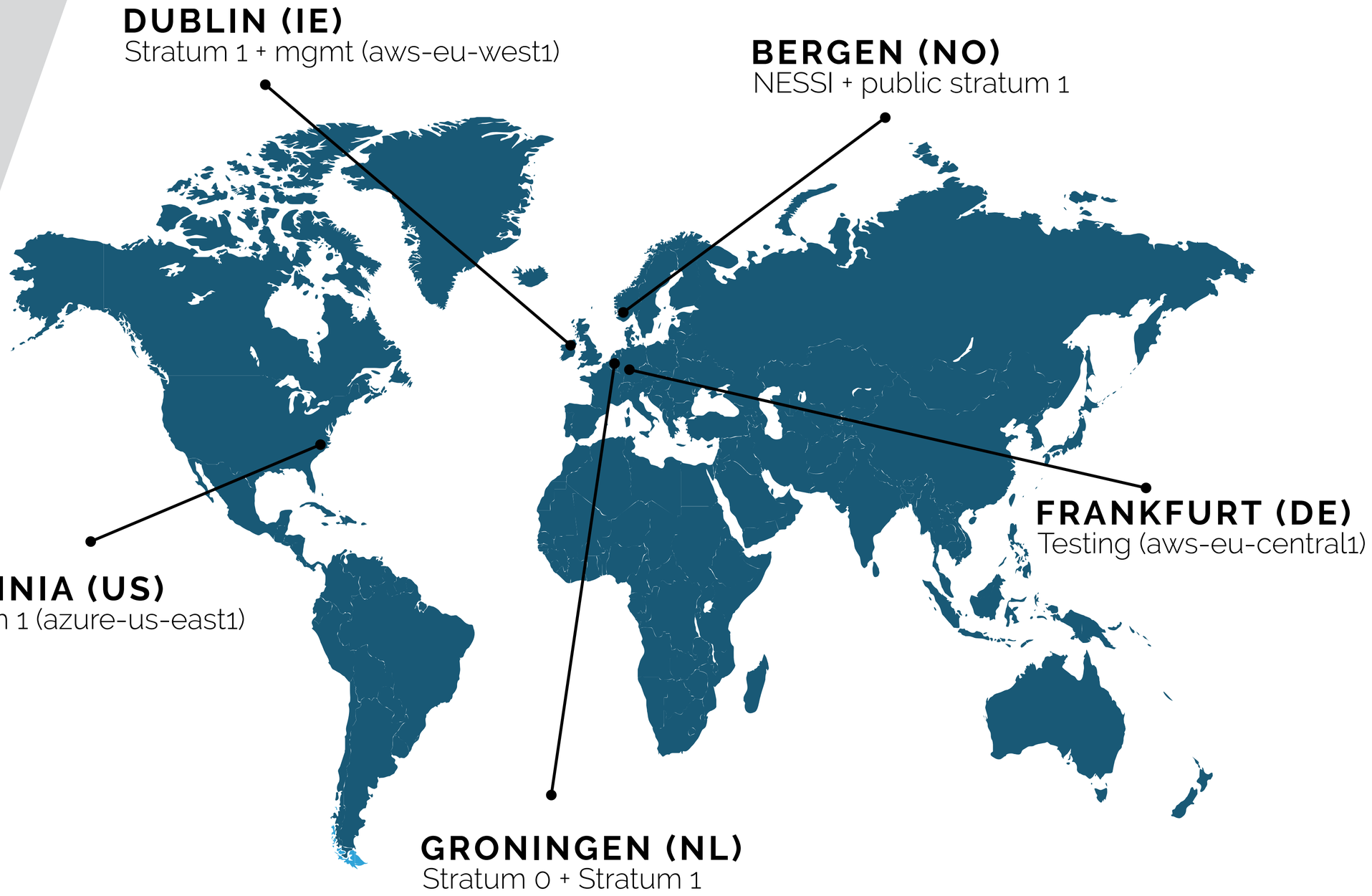
Terje Kvernes | 2022-09-15

ANYWHERE, ANYTIME

Where are we today?

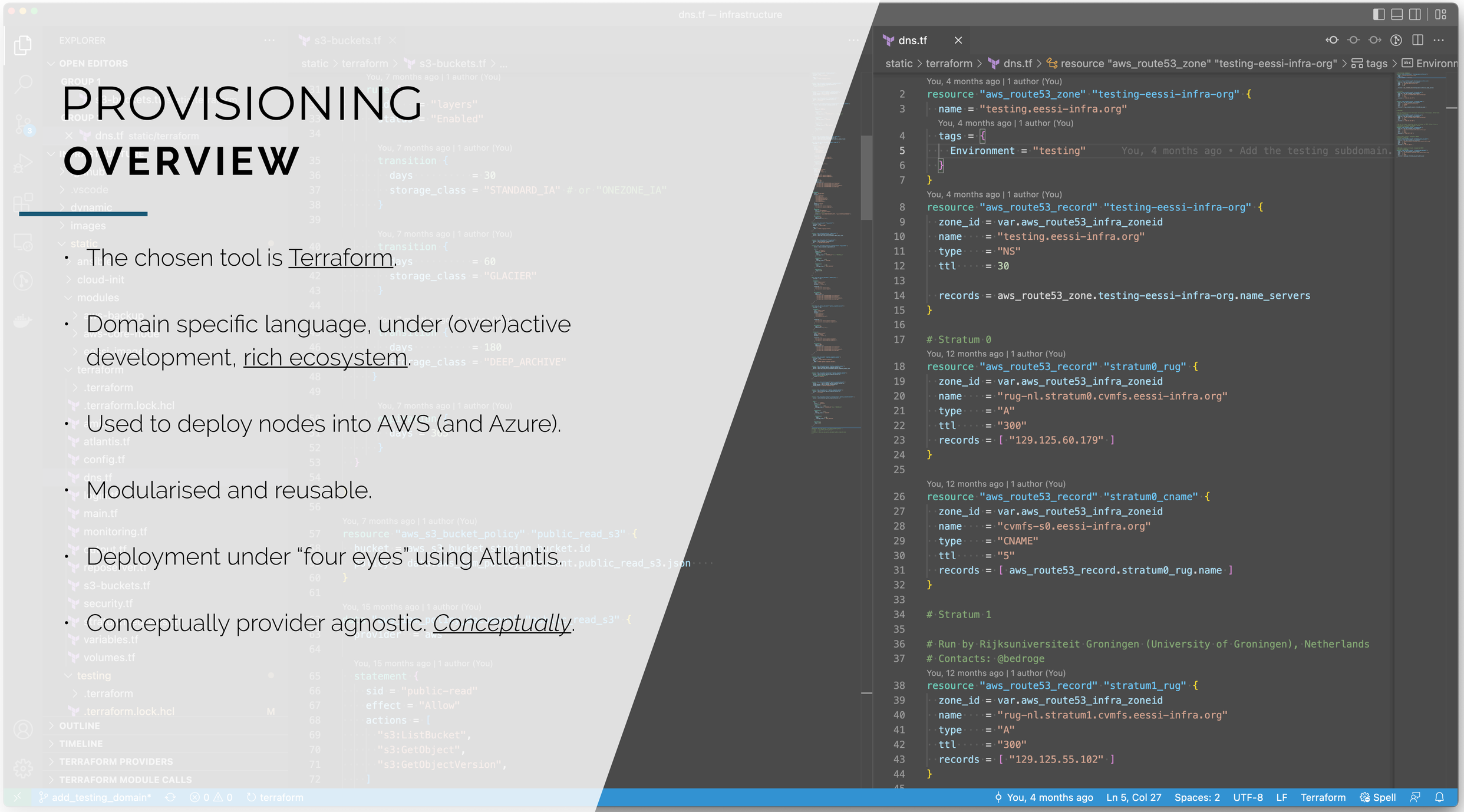
Multiple providers
Multiple locations

And more to come…



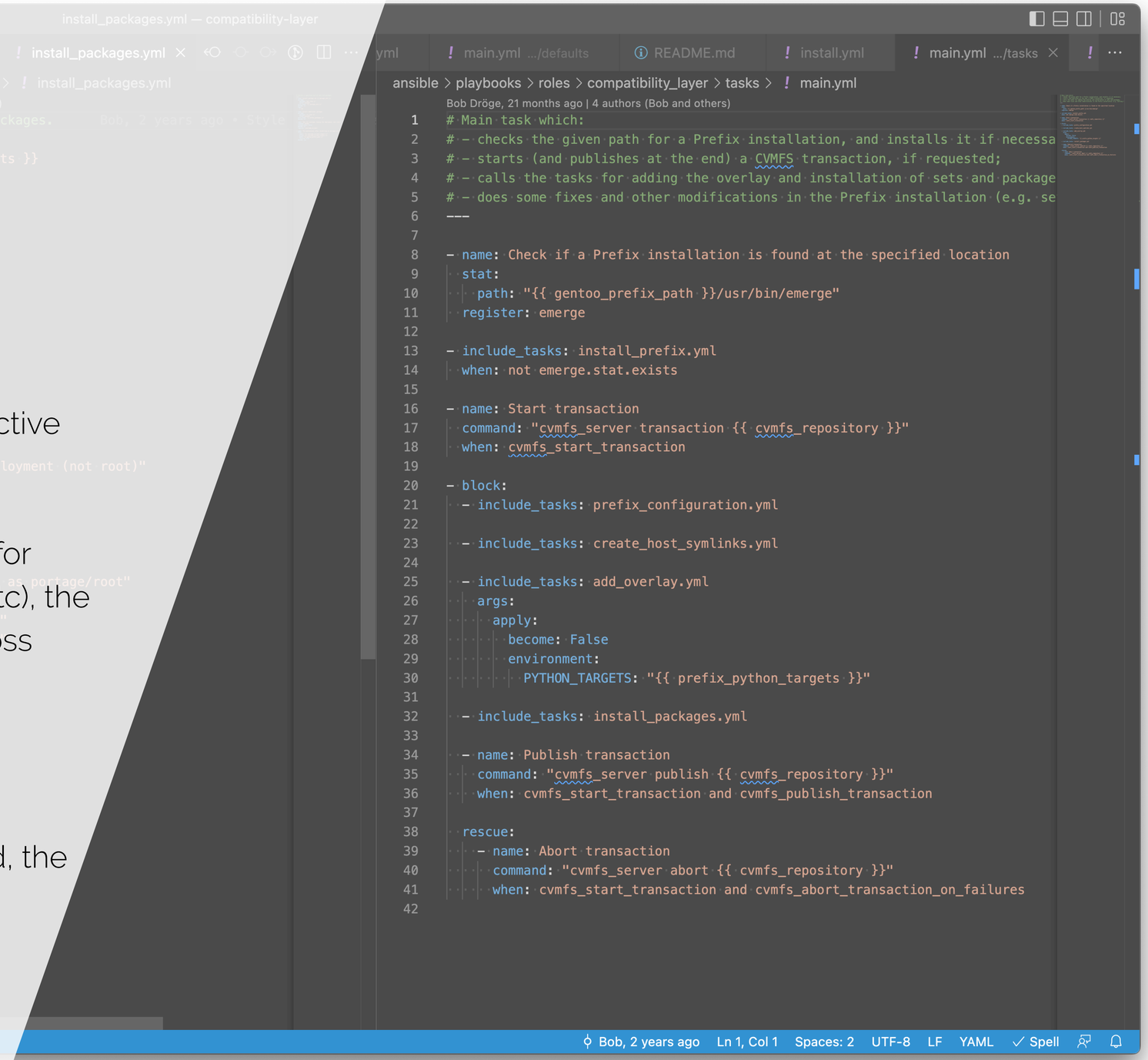
PROVISIONING
OVERVIEW

- The chosen tool is Terraform.
- Domain specific language, under (over)active development, rich ecosystem.
- Used to deploy nodes into AWS (and Azure).
- Modularised and reusable.
- Deployment under “four eyes” using Atlantis.
- Conceptually provider agnostic. *Conceptually*.



CONFIGURATION
OVERVIEW

- The chosen tool is Ansible.
- YAML-based, human “readable”, under (over)active development, rich ecosystem.
- Due to repositories being layer-based (a repo for software layer, another for the combat layer, etc), the ansible scripts themselves are distributed across *many* repositories…
- No central playbook deployment or structure.
- (In what repo do I find, or would I expect to find, the ansible scripts to set up a client?)



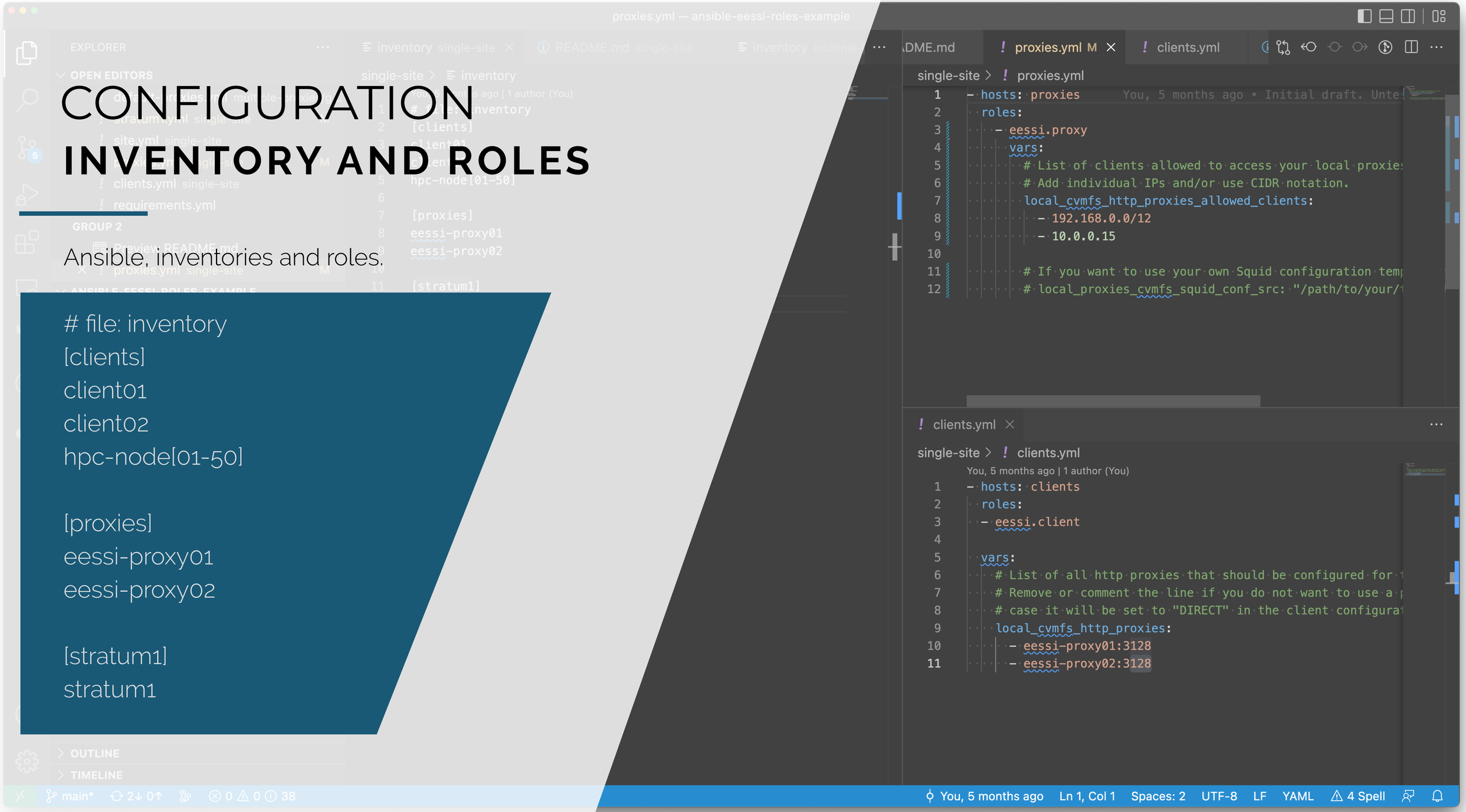
CONFIGURATION
INVENTORY AND ROLES

Ansible, inventories and roles.

```
# file: inventory
[clients]
client01
client02
hpc-node[01-50]

[proxies]
eessi-proxy01
eessi-proxy02

[stratum1]
stratum1
```



CREATING INFRASTRUCTURE

PLANNING
DEFINE NEEDS

What infrastructure is needed? Virtual machines? Storage buckets? What access rules are required? Cost/benefit analysis, and so on.

PROVISION
TERRAFORM

Produce terraform code, test directly against the test environment, push PR to a branch in the infrastructure repo on GitHub to have Atlantis test the code. Repeat until Atlantis stops complaining.

APPROVE
GIT+ATLANTIS

Approving terraform code to have it applied is done by approving a PR in the infrastructure repo. Atlantis then executes the changes.

INVENTORY

GIT

Add to or create a new infrastructure inventory. As of now this is a manual task.

CONFIGURE

ANSIBLE

Run the playbooks to deploy the configuration. Currently performed manually. A long term goal is to have a master deployment node.

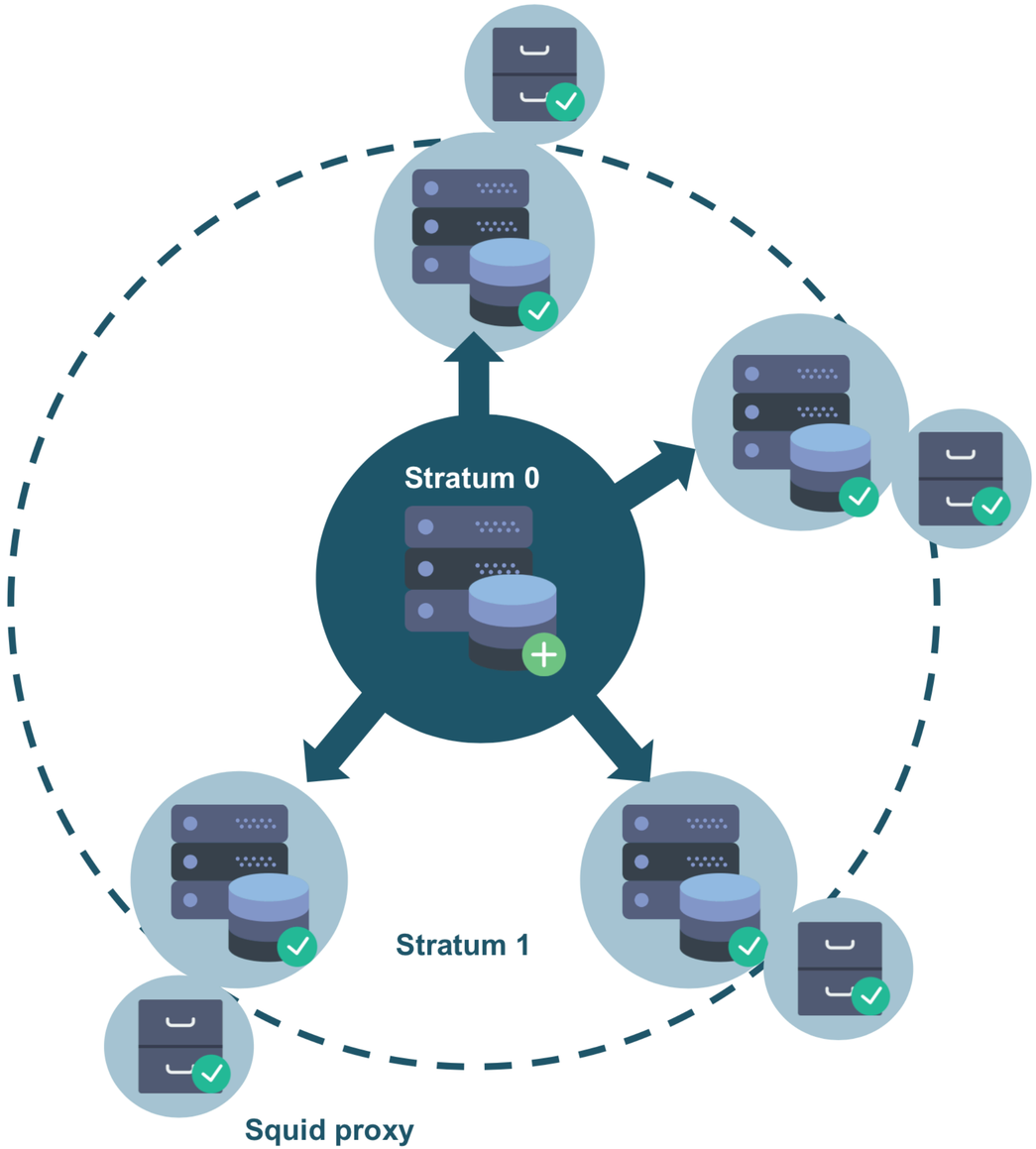
VALIDATE

MONITORING

Validate the changes by adding the new services to the monitoring stack.

WHAT'S IN AN EESSI?

A familiar high-level overview



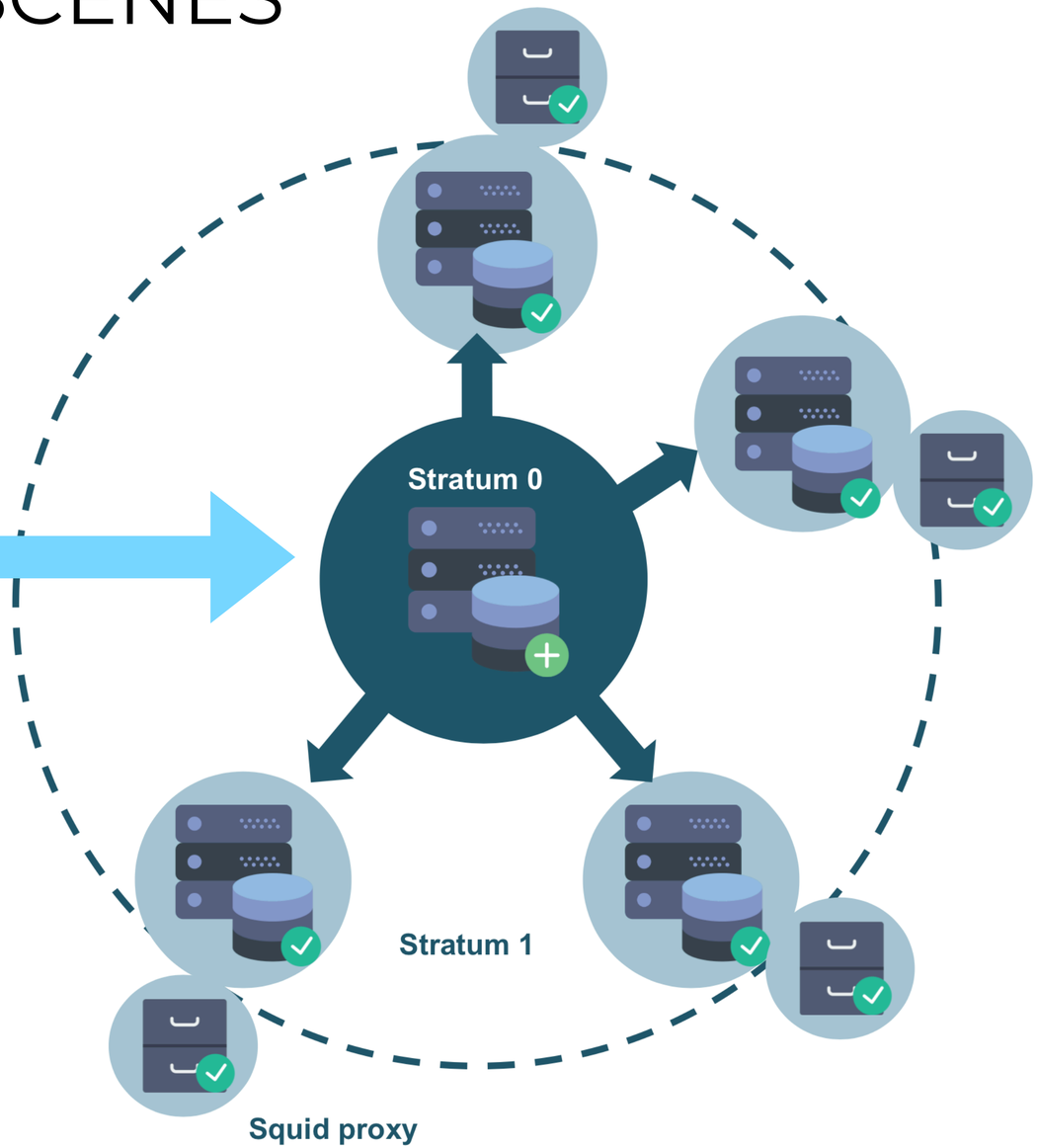
(Ikoner fra https://www.flaticon.com/authors/smashicons)

EESSI : BEHIND THE SCENES

STRATUM 0

Stratum 0:

- Central server
- Unique
- Hosts the CVMFS volumes
- Mostly automated day-to-day
- Extremely limited access
- May be provisioned via terraform
- Partial configuration management via Ansible

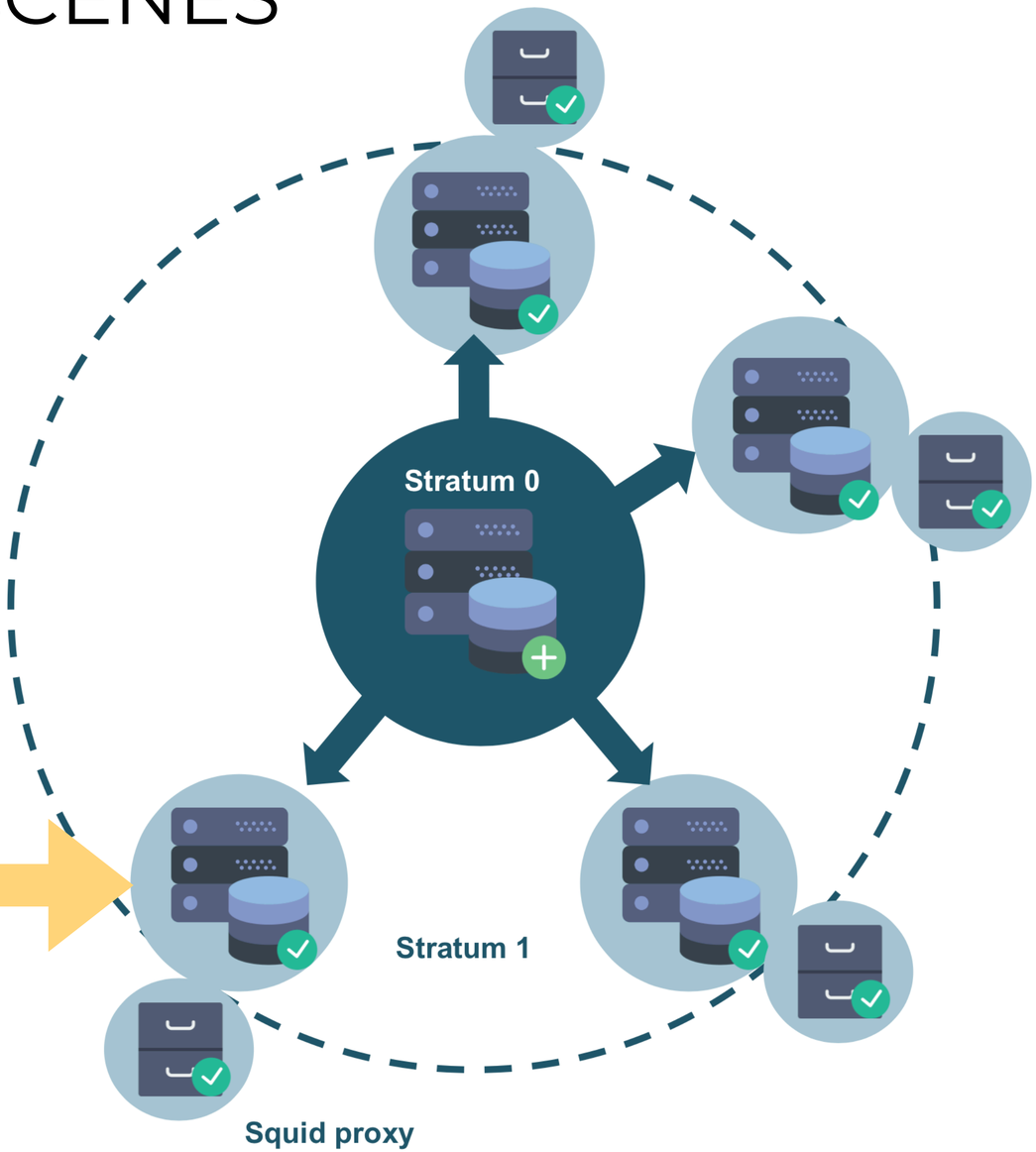


(Icons from https://www.flaticon.com/authors/smashicons)

EESSI : BEHIND THE SCENES

STRATUM 1

- Replicates stratum 0
- Complete copy of volumes
- Serves data read-only
- A number of these worldwide
- Geographically distributed
- Runs a standard webserver
- Reduces load on stratum 0
- Offers redundancy
- Provisioned via Terraform
- Configured via Ansible

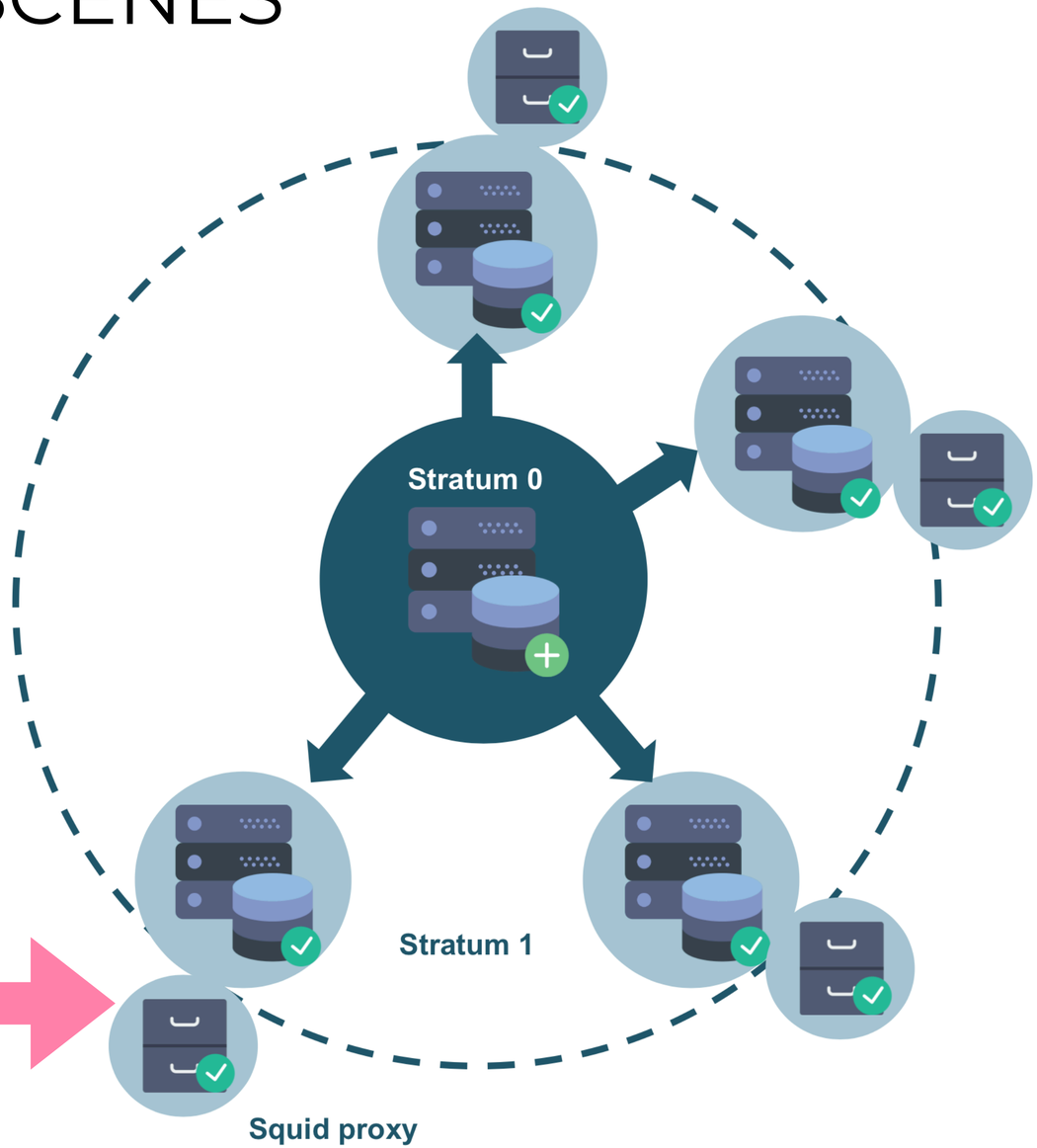


(Icons from https://www.flaticon.com/authors/smashicons)

EESSI : BEHIND THE SCENES

PROXY

- Reverse proxy for stratum 1
- I/O cache for clients
- Improved user experience
- Load balancing is an option
- Primary contact point for clients
- Lots and lots everywhere
- Provisioned via Terraform
- Configured via Ansible

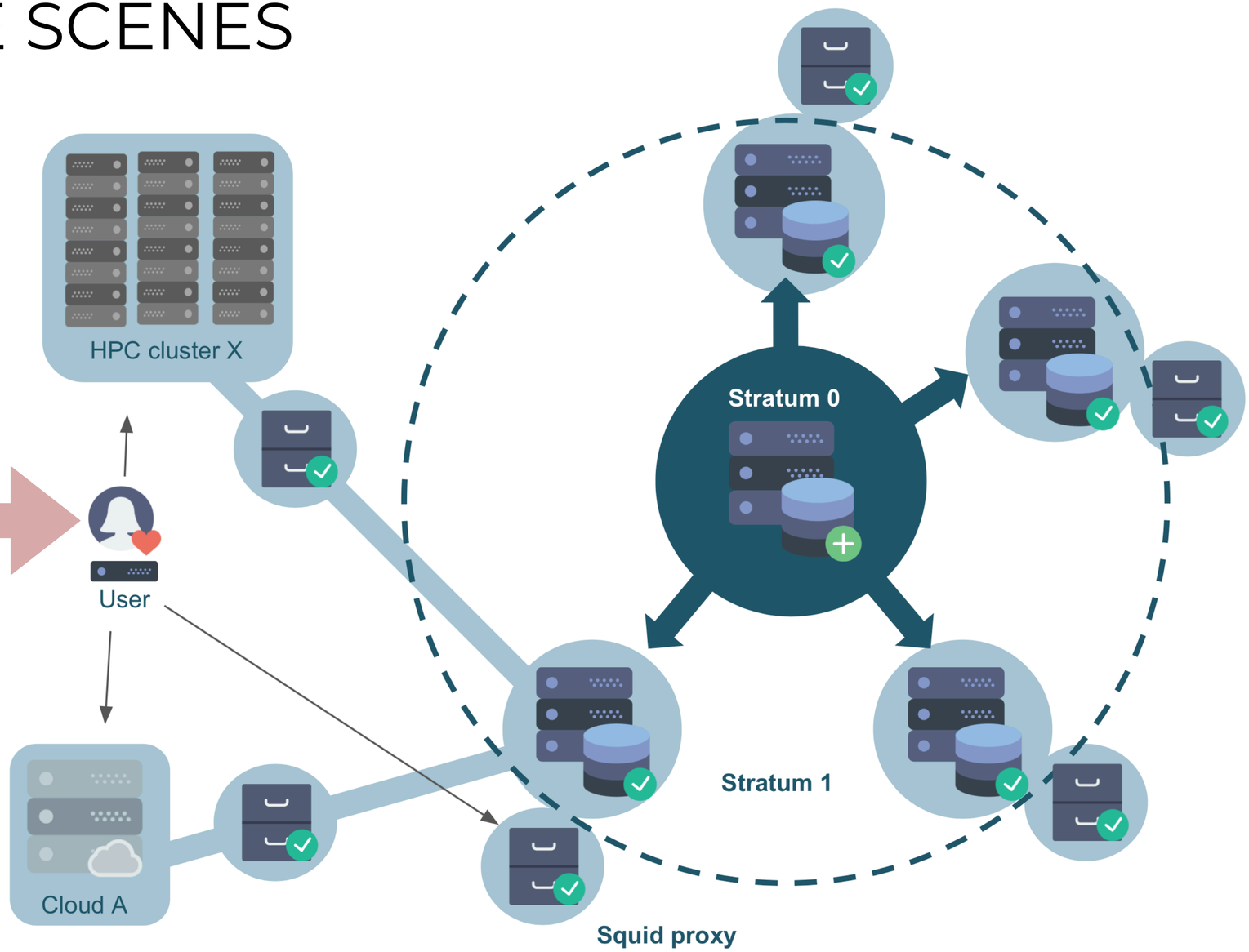


(Icons from https://www.flaticon.com/authors/smashicons)

EESSI : BEHIND THE SCENES

CLIENTS

- Fetches software from a squid proxy or a stratum 1
- Laptops, workstations, HPC-clusters, cloud machines, etc
- Private, personal, or managed devices
- A local filesystem cache provides performance
- All clients experience the same EESSI software stack everywhere!
- Can be configured via Ansible



(Icons from https://www.flaticon.com/authors/smashicons)

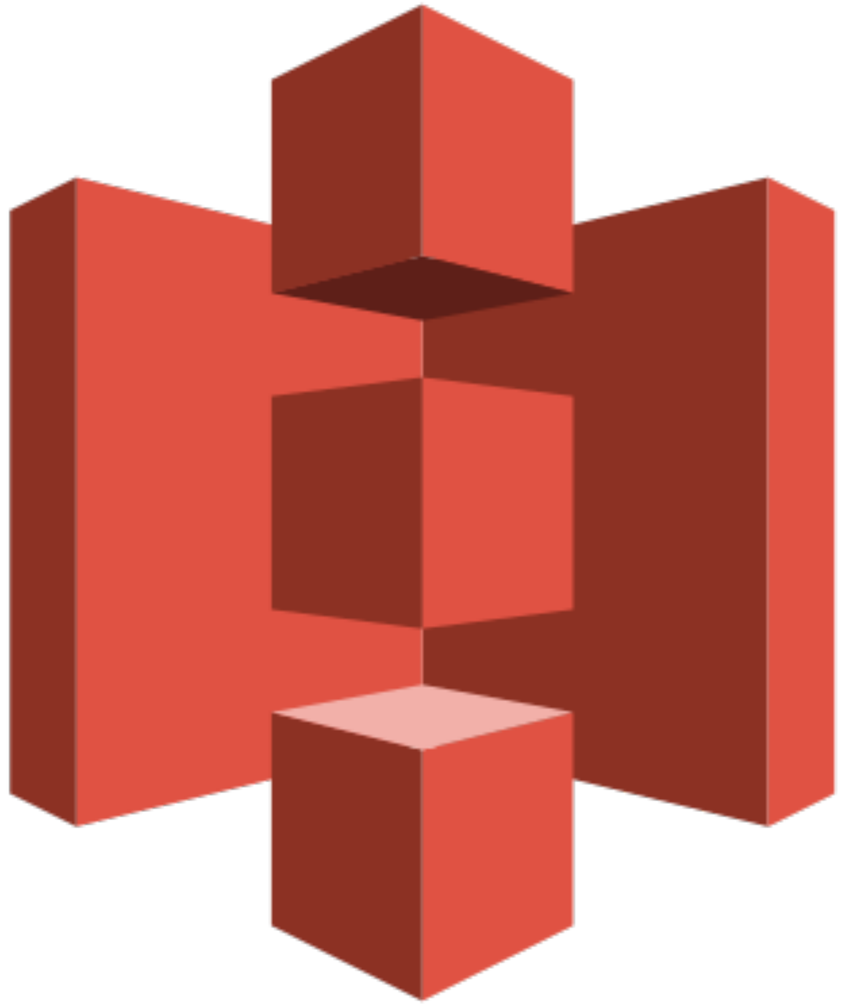
EESSI : BEHIND THE SCENES
OTHER STUFF

- Monitoring (prometheus + grafana)
- Atlantis (deployment)
- Login node with local persistent storage
- S3 buckets (compatibility layer + software layer tarballs, interaction with GitHub tools, logging, and more)
- Status page
- Ephemeral nodes

- Identity providers
- Access control (hosts, services, networks, users, roles, groups…)
- …

OTHER STUFF
S3 STORAGE

- Provisioned and access controlled via terraform
- Used for staging tarballs, gentoo snapshot backups, and logging
- Currently uses AWS as its provider
- Lots of lifecycle management baked into the terraform modularisation
- The ACL environment is complex, allowing users or specific nodes or specific tokened nodes access.

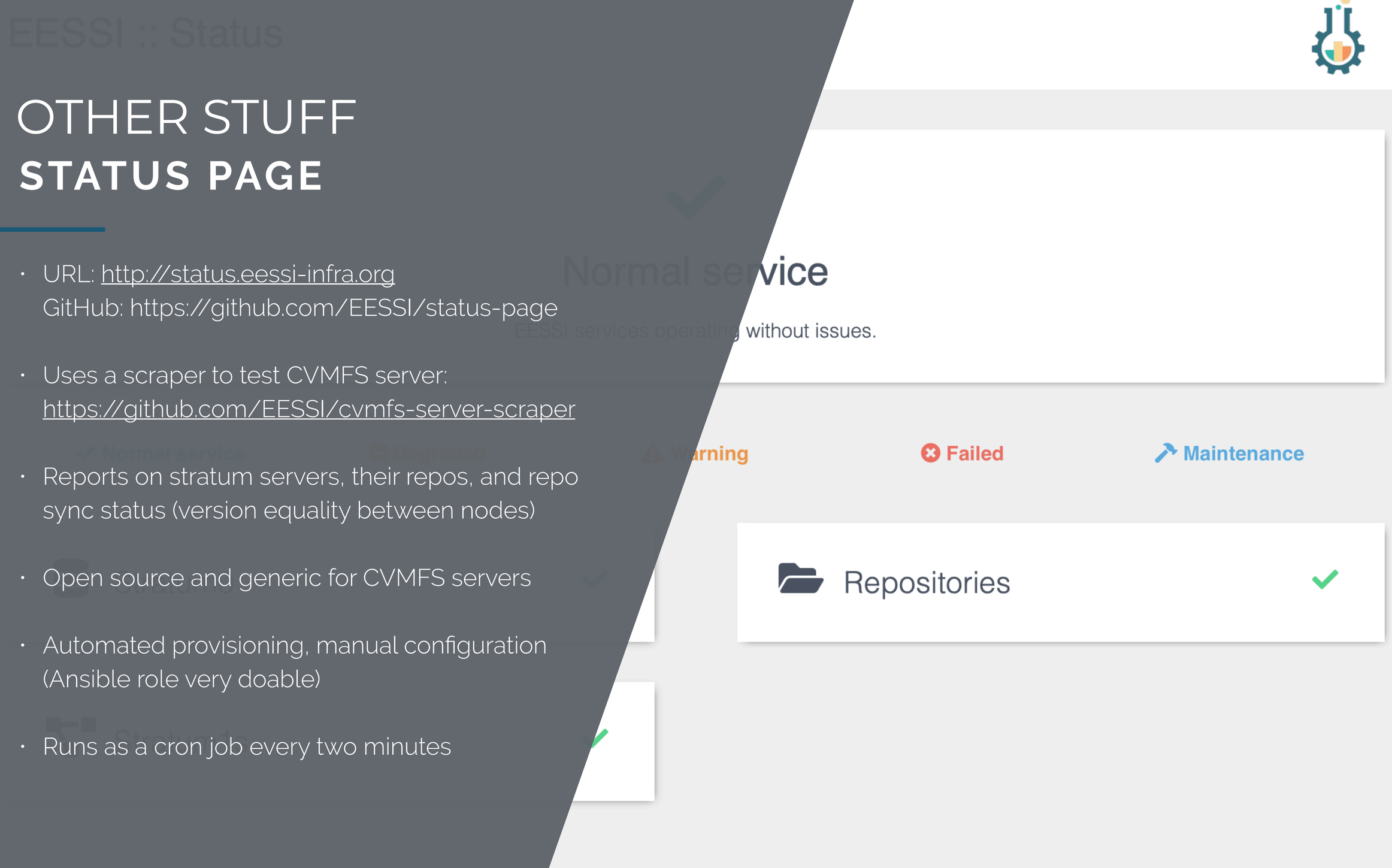


OTHER STUFF
CITC

- Cluster in the Cloud
- Build/test/deploy EESSI-related “stuff”
- Good for hackathons
- Automatic scale-out (spins up worker nodes on demand)
- Off the shelf software, customisations

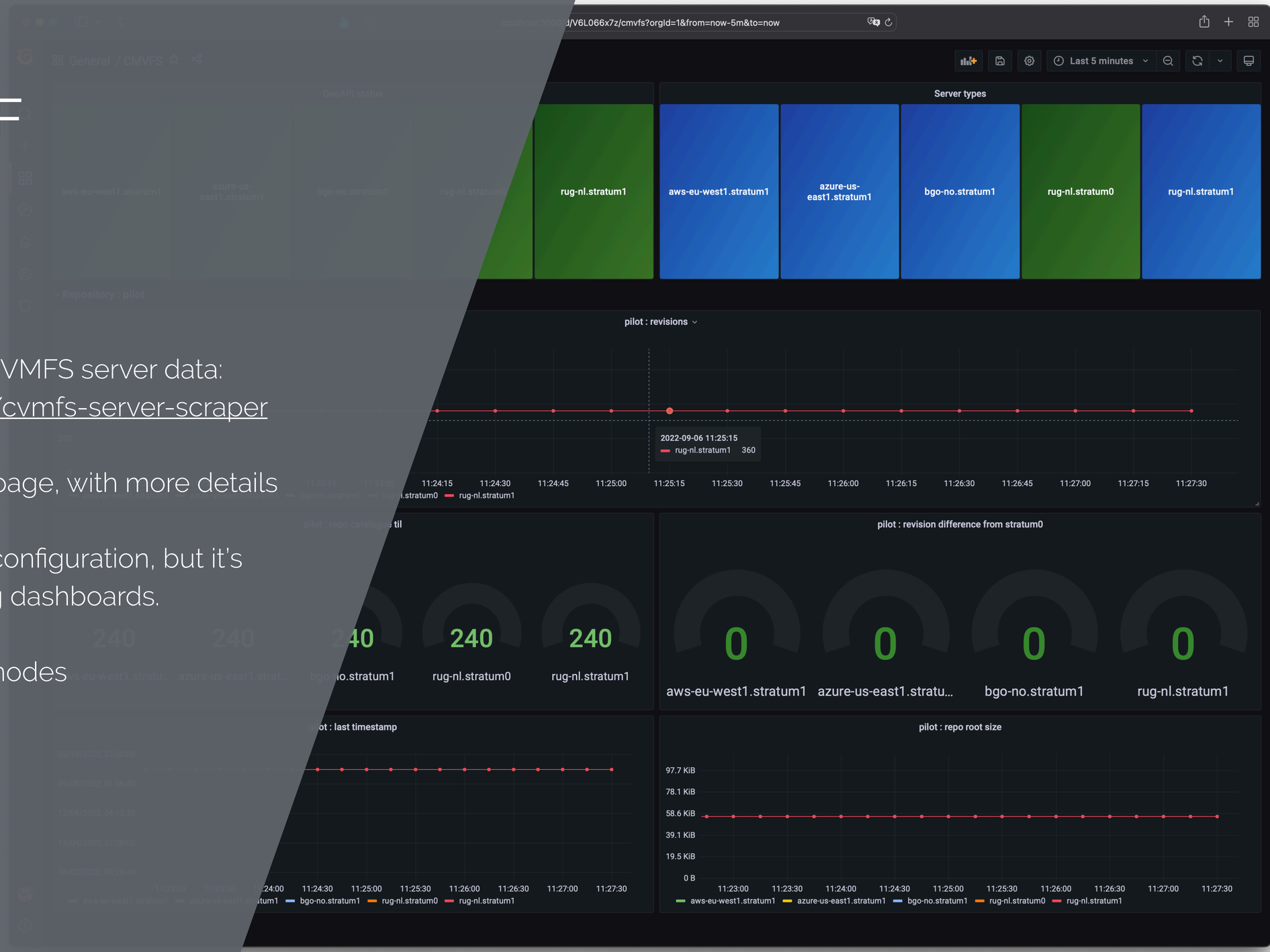
OTHER STUFF
STATUS PAGE

- URL: http://status.eessi-infra.org
  GitHub: https://github.com/EESSI/status-page
- Uses a scraper to test CVMFS server:
  https://github.com/EESSI/cvmfs-server-scraper
- Reports on stratum servers, their repos, and repo sync status (version equality between nodes)
- Open source and generic for CVMFS servers
- Automated provisioning, manual configuration (Ansible role very doable)
- Runs as a cron job every two minutes



OTHER STUFF
MONITORING

- Prometheus + Grafana
- Uses a scraper to collect CVMFS server data: https://github.com/EESSI/cvmfs-server-scraper
- Historic view of the status page, with more details
- Veeeeeeeeeeeery manual configuration, but it's actually solvable, including dashboards.
- Only monitors production nodes



THE FUTURE

PLANS

EESSI : BEHIND THE SCENES
CURRENT PLANS : STRATUM SERVERS

- New physical stratum0
- Use security devices (yubi-key) to approve signing keys rotated regularly
- Admin access to all Stratum servers based on GitHub team membership (except maybe stratum0)
- Locked down default OS image and access control for all stratum servers
- Reconsider image creation (Packer) and instead configure very basic images
- World-wide public Stratum1 service, utilising AWS, Azure, and possibly other providers
- Automated monitoring of public stratum1 via scrapers

EESSI : BEHIND THE SCENES
CURRENT PLANS : OTHER STUFF

- Migration to eessi.io as our primary domain
- Further work to unify access control across services (github tasks/stratum servers/monitoring/atlantis)
- Dedicated management node with node-based access to relevant infrastructure
- Clean up Ansible playbook structure, ensure regular deployment
- Better CD/CI pipelining for playbooks
- Centralize critical meta data (what are the public stratum1s etc) and use this data *everywhere*
- Try to stabilise the infrastructure code base, possibly a clean slate?

THE FUTURE

CHALLENGES

EESSI : BEHIND THE SCENES
CURRENT CHALLENGES : CODE BASE

- There are many ways of making things work
- Terraform and Ansible have best practices
- Following them is a good thing
- You won't be following them
- But you can try
- Versions change, code needs to change with it
- Nothing rots like infrastructure code

EESSI : BEHIND THE SCENES
CURRENT CHALLENGES : MONITORING

- Monitoring needs further automation based on an authoritative source of public stratum1s
- How do we tag resources for severity?
- When do we alert?
- Why do we alert?
- Who do we alert?

(Also, how do we alert in this day and age? E-mail? Slack? SMS? For what severity? For what security? How do we inform end users?)

EESSI : BEHIND THE SCENES
CURRENT CHALLENGES : TESTING

- Testing environment needs work
- Stratum1 isn't trivial to test (size, time, etc)
- A complete mirror of prod is possible but will carry a financial and logistical cost
- It's never going to be a perfect mirror
- What do we do with login nodes, management resources, log buckets, etc?
- Ideally we monitor the testing environment as if it was prod but report things in different channels
- This is one of the big outstanding bits of the current environment

EESSI : BEHIND THE SCENES
CURRENT CHALLENGES : DESIGN

- We *need* multi-provider redundancy for EESSI infrastructure
- IaaC is nice, but it isn't enough
- Most provisions are automated
- Lots of configuration isn't automated, even if it is scripted
- The solution is to move everything to "GitOps"
- Atlantis (and/or Terraform Cloud? Github actions?) for provisioning
- Github actions, Semaphore or Rundeck for Ansible?
- If we run self-hosted services we need redundancy with load balancers with health checks in front (ELB)

LESSONS

Infrastructure is a living thing.

Infrastructure is never complete.

Automation isn't.

Monitoring is hard. Reporting is harder.

Testing and production are sometimes similar, but never the same.
(And if it is, one has other problems.)