MASTERCLASS

INFRASTRUCTURE AS CODE

Virtualization Conference

SINISA

CTO

Learning how to develop

BENEDIKT

Developer

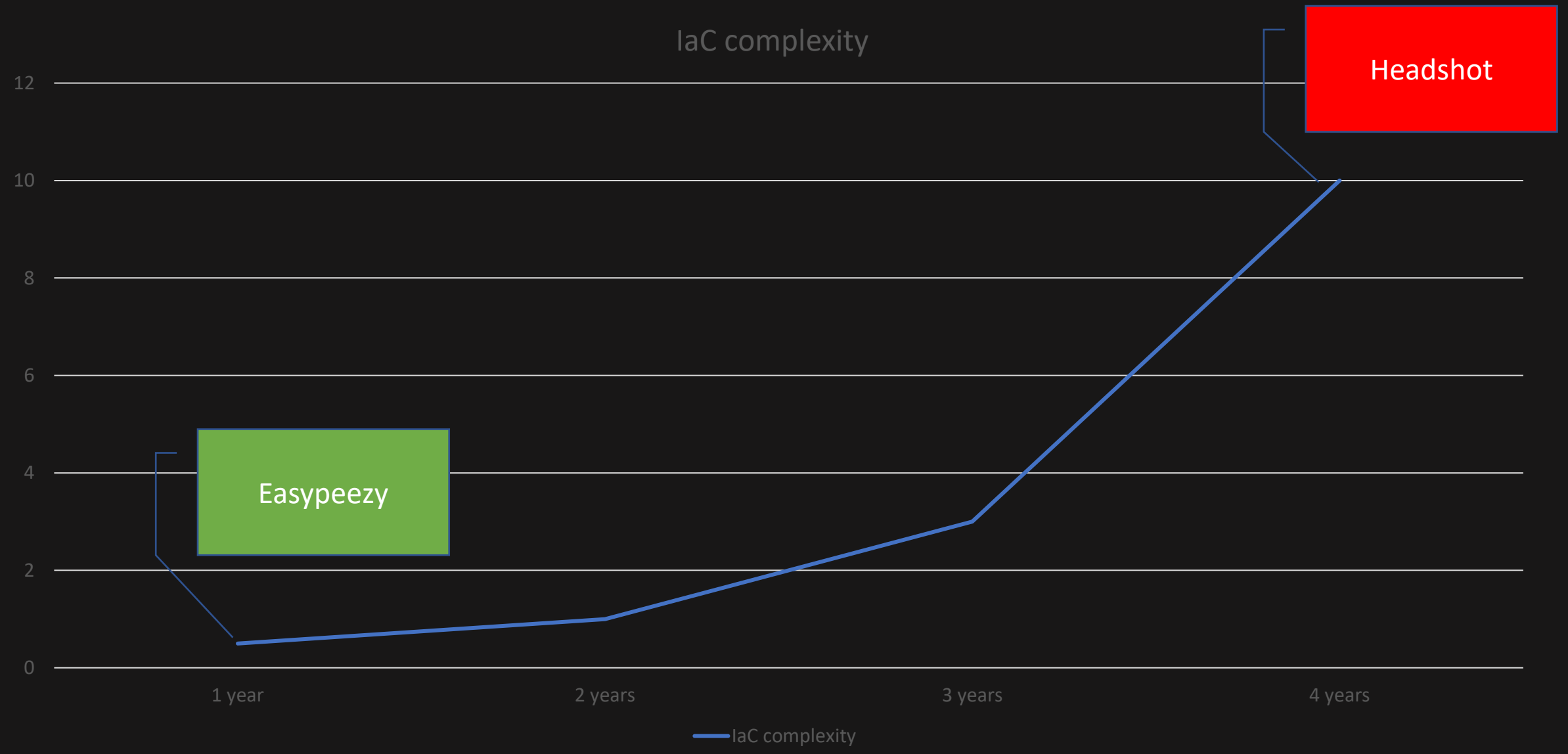
Learning how to do Infrastructure

Virtualization Conference

WHY THIS MASTERCLASS?

- Learn to live with the complexity
- Writing Terraform Code is not enough
- Get an understanding of the dependencies (explain this to your CEO)
- Understand the organizational changes you have to make
- Get a rocket start for your own implementation

You´ll never be the same after implementing IaC.



Virtualization Conference

How to survice IaC?

The story about pain and getting to a point where it doesn´t hurt anymore.

- Development team -

AGENDA

Part I

- Organizational stuff around IaC

Part II

- IaC with Terraform

Part III

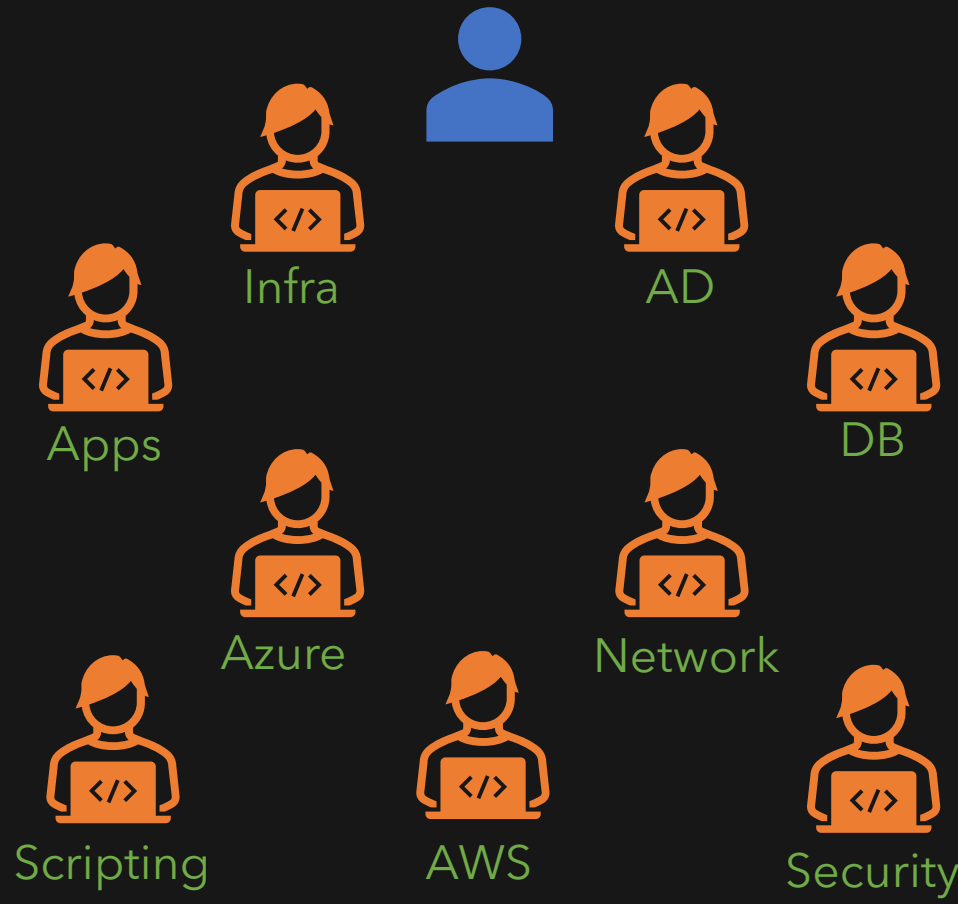
- - Building the pipeline

PART I

Virtualization Conference

SCRUM

- Principles
  - Commitment
  - Courage
  - Focus
  - Openness
  - Respect
- Roles
  - Product Owner
  - SCRUM Master
  - Development Team



Implementing IaC is always a complete change from ITIL based organizations to agile work forms.

PRIORITIZATION

MoSCow method

- Must Have
- Should have
- Could have
- Won´t have (this time)

Priority	Classification	Description	When to fix
1	Critical defect	The defect affects critical functionality or critical data. It does not have a workaround	immediately
2	Major defect	The defect affects major functionality or major data. It has a workaround but is not obvious and	Allign with team, same Sprint or next is difficult
3	Average defect	The defect affects minor functionality or non-critical data. It has an easy workaround	Next Sprint
4	Minor defect	Cosmetic defects	When time allows

DAILY

Timebox: 15 Minutes

Participants in daily Scrum: Development team

Questions to be answered by every team member:

- What have I worked on since last Daily?
- What am I going to work on until next Daily?
- What is currently blocking me? (Is this a team internal blocker, or Cross Team?)

DAILY WORK I/II

- Enable your camera so your colleagues can see you
- Attend the daily at 09:00 AM
  - Adjust all your Work Items with the current status beforehand
- All of the code has to be pulled before starting to work on an Item.
- All of the code has to be pushed by EOB to the corresponding branches or repositories.
- Create Feature Branches for every Story you start working on.
- No uncommited code on the devices
- Be polite
- Be on time to meetings

DAILY WORK II/II

- Use Teams for 1:1 communication and the different channels if you have a question one might answer.
- Don´t bother colleagues with unnecessary chats during the day. They are working hard to get their work items done.
- Use Google before you ask a question.
- Don´t push code into MASTER (it shouldn’t be possible - if yes let your colleagues know)
- Each member of the team knows what the others are doing (if you don’t know how will you add value to the team?)

Other Day to Day activities

- Make sure that the pipeline is running smoothly
- Co-ordination and collaboration is the key for DevOps
- Work on Automation Backlog, Automation is soul of DevOps
- Infrastructure Management
- Dealing with Legacy stuff
- Exploration
- Removing bottleneck
- Documentation
- Training and Self Development

DEFINITION OF DONE (SIMPLE)

- Story can be released
- Story is integration tested
- Release Notes updated
- Customer Acceptance Test successful
- Technical documentation is updated

ESTIMATION

Backlog Item	Story Points	Level	Explanation
Task	0,5	Sleep	We can finish this task during our sleep
Task	1	Childsplay	We need to get up, but actually it is not a lot more work than that
Task	2	Peanuts	Not for Children. For experience developers this is no challenge.
Task	3	Rocket	This is a sophisticated task. But no rocket science.
Task	5	High jump	A task that can be handled with good planning and preparation
Story	8	Thermometer	Even experienced developers start sweating when tackling this task
Story	13	Fire	Careful - a task complexity where others already burned their fingers
Feature	20	Bullfight	The task is a beast. It can only be tackled with determination and focus
Feature	40	Mammoth	A huge task. It seems impossible to tackle. With a great degree of teamwork, and proper preparation this can be handled.
Epic	100	Iceberg	From the distance it looks nice, but if you come closer and dig deeper you realize that most of the effort is yet unknown. Very dangerous.

PRODUCT BACKLOG

- The Product Backlog is an ordered list of everything that is known to be needed in the product
- A Product Backlog is never complete
- Requirements never stop changing, so a Product Backlog is a living artifact
- Refinement usually consumes no more than 10% of the capacity of the Development Team

ROADMAP

- Start with big picture
- Identify big items
- Break it down and fill the Backlog
- Estimate how long does a task take
- Generate Roadmap and Requirements
- Share with team and validate
- Keep improving

SPRINT

The heart of Scrum is a Sprint, a time-box of one month or less during which a "Done", usable, and potentially releasable product Increment is created.

- No changes are made that would endanger the Sprint Goal
- Quality goals do not decrease
- Scope may be clarified and re-negotiated between the Product Owner and Development Team

SPRINT PLANNING

Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint.

Answers the following:

- What can be delivered in the Increment resulting from the upcoming Sprint?
- How will the work needed to deliver the Increment be achieved?

SPRINT REVIEW

A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed.

The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint.

WORK ITEM TAGGING

Tags can be used to mark User Stories, Tasks and Epics for different use cases.

Grooming

- These items should be groomed in the next Grooming Session

Design

- These items are still in a design phase and not yet ready for the Sprint

DOCUMENTATION

Write all of your technical documentation in **Markdown**.

- Store it in git
- You get versions and change tracking
- Automatic docs build with e.g. Hugo
- If you need diagrams use draw.io

https://www.markdownguide.org

https://app.diagrams.net

CHANGELOG

Types of changes

- **Added** for new features.
- **Changed** for changes in existing functionality.
- **Deprecated** for soon-to-be removed features.
- **Removed** for now removed features.
- **Fixed** for any bug fixes.
- **Security** in case of vulnerabilities

https://keepachangelog.com/

VERSIONING

Software versioning is the most widely adopted scheme for assigning unique version version numbers to software releases.

Practice Semantic Versioning for all projects.

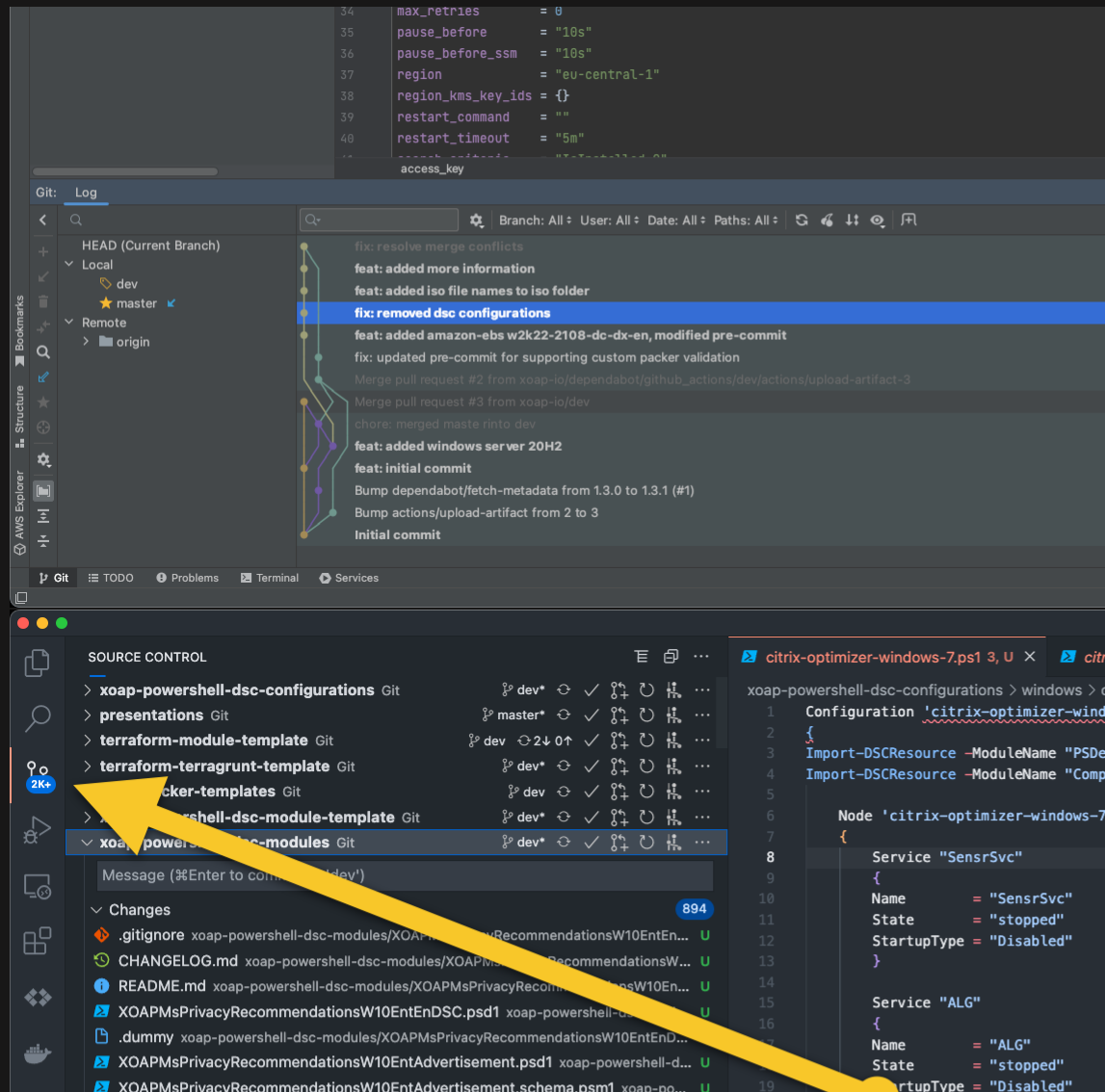
Using this versioning standard helps to reduce the entropy related to Dependency Hell.

https://semver.org/

MANAGE CODE

- Azure DevOps repos
- GitHub private and public repos
  - GitHub Desktop
  - Visual Studio Code
  - IntelliJ IDEA

- Much more out there…
  - Bitbucket
  - Gitlab

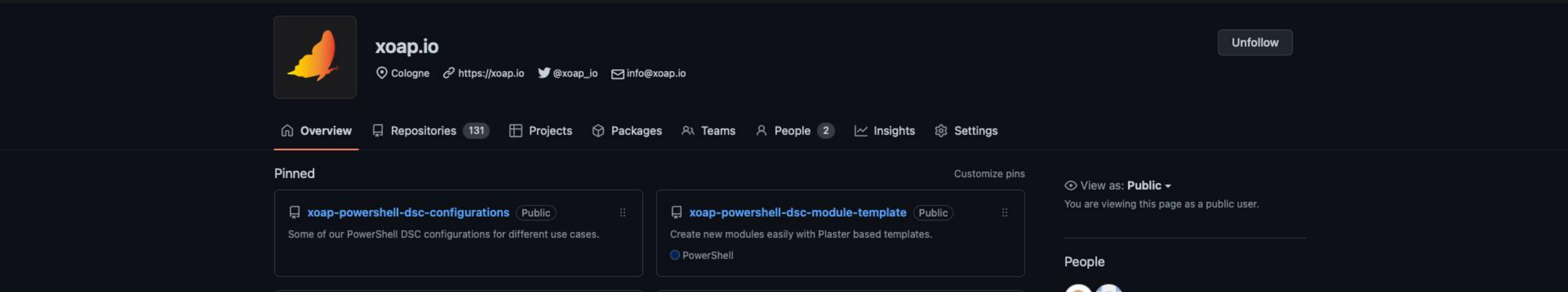


GITOPS

Git as the single point of truth!

This means:

- Pipelines in git
- Code in Git
- Documentation in Git



Why we are moving to GitHub?

- More Actions available
- Better Terraform support
- Better integration of public registries

COMMUNICATION

Microsoft Teams

- Video Conferencing, Chat
- Subchannels for different topics, documents, links
- Notifications for pipelines and changes
- Update subscriptions
- Monitoring of Infrastructure

Microsoft OneNote

- Sprint Planning, Meeting Notes

Azure DevOps Boards

- Plan and assign Stories and Tasks
- Documentation: Link commits to Stories and Tasks

Virtualization Conference

NEXT PART II

5 Minutes bio-break

PART II

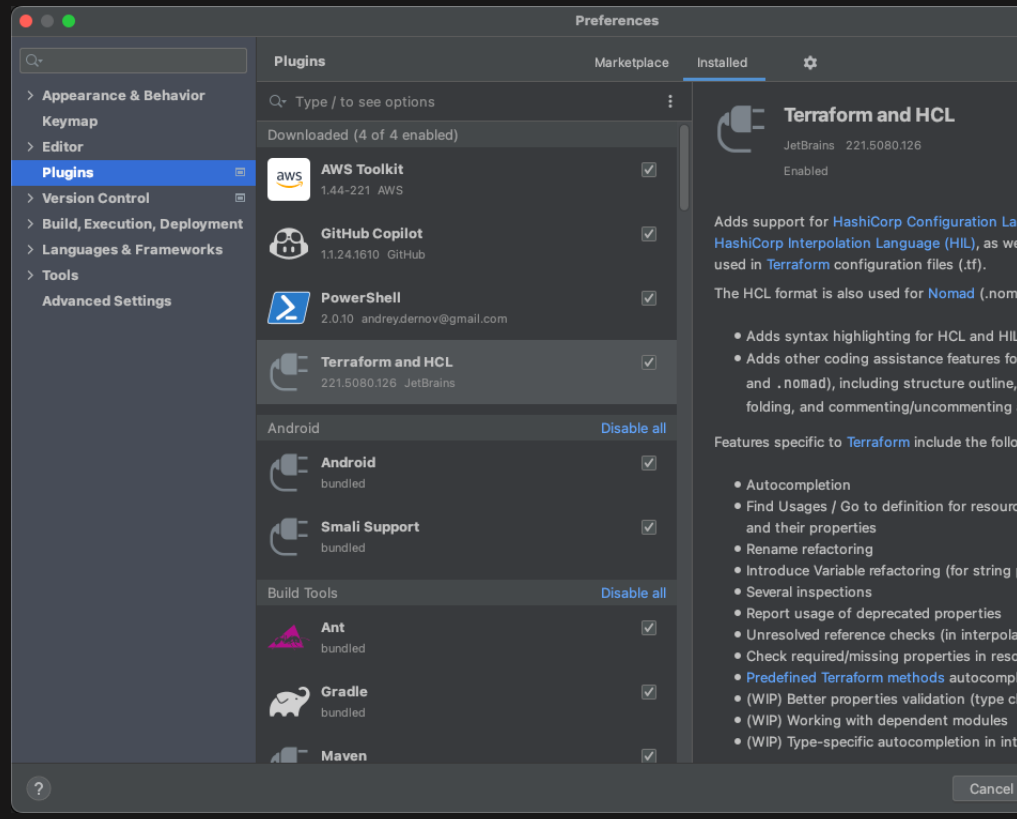
- Tools
  - IDE
  - Extensions

Azure DevOps

- Boards
- Repos
- Pipelines

IDE

- Visual Studio Code
  - More customization
  - Less “Intellisense”
- IntelliJ IDEA or PyCharm
  - Better Terraform support
  - More powerful but also more complex
- Register for GitHub Copilot Preview



Virtualization Conference

Other Tools

- Ohmyzsh (for all the Mac or zsh users)
- tfupdate
- pre-commit
- AWS-vault
- Terracognita (infra reverse engineering)

TERRAFORM

- Why
- How
- Modules
- Stacks
- Extensions

Terraform – Why?

- Improved multi-cloud infrastructure deployment
- Infrastructure as code (IaC)
  - Stateful resources
  - Change tracking
  - Automated testing
  - Enforced tagging and naming
- Reusability with modules and stacks
  - Reduced development costs
  - Reduced time to provision
- No „click-ops“
  - Staging with prod, acc, dev through configs
  - Less error prone

Terraform – How?

Terraform – How?

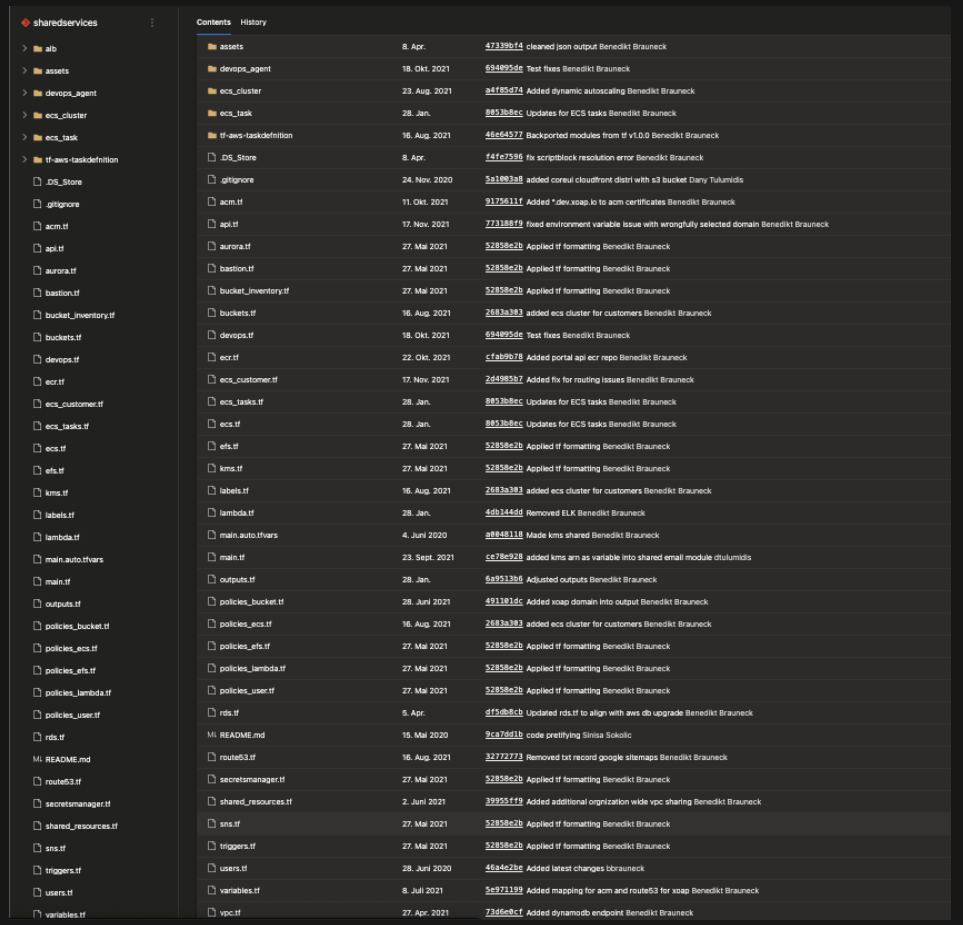
- Main.tf
  - creates ressources
- Output.tf
  - human readable output of created resources
- Variable.tf
  - input variables for configuration
- Provider.tf
  - configuration for provider i.e. Azure, AWS, Alibaba, etc.
  - Statefile configuration
- Versions.tf
  - required versions of providers and their source location

Terragrunt – Why?

- Orchestration around Terragrunt
- DRY processes
  - Provider initialization
  - Statefile management
  - CLI hooks
- Better project structure
- „Deployment logic“ / dependency management
- Decoupling configuration from resources
- Easier staging
- Easier linking of values between stacks

Terragrunt – Why?

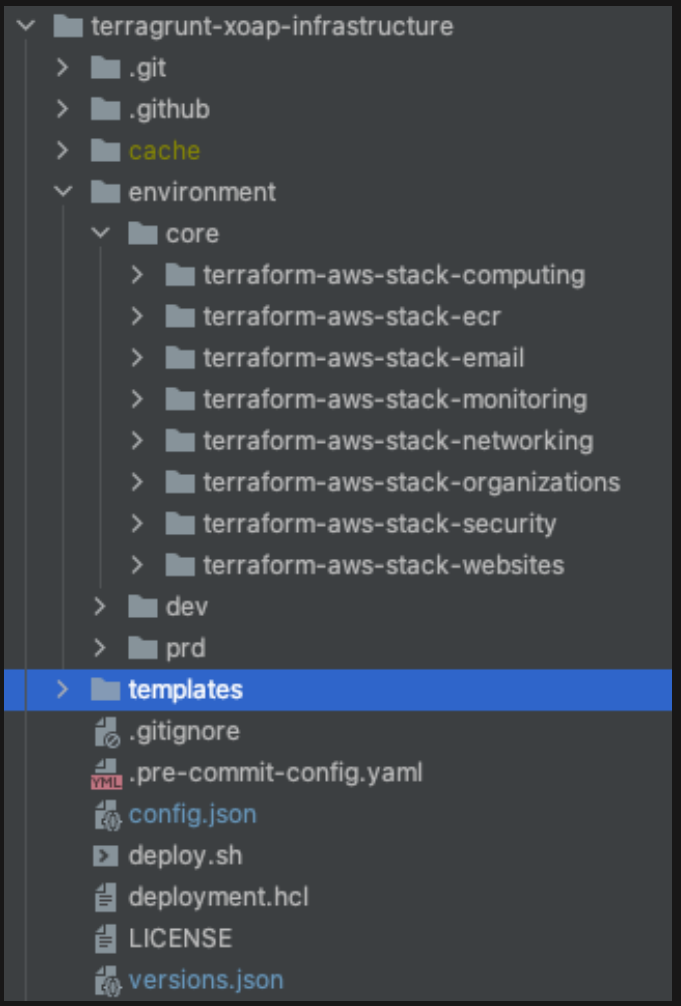
- Projects always extend with time
- Features will be added/removed
- Some Services are hard to update (i.e. VPC)
- Migration paths are difficult
- Provider compability issues
- Big Stacks are SLOW



Virtualization Conference

Terragrunt – Why?

- Automatic initialization
- Easier migration paths
- Reduced update amount (targeted)
- Reusability
- Flexibility
- More parallelism (especially in multi-cloud setups)
- Better deployment flow
- Less hacky stuff (export configs for other Stacks)
- Source mapping (local development vs. pipeline deployment)



Deployment old

Deployment new

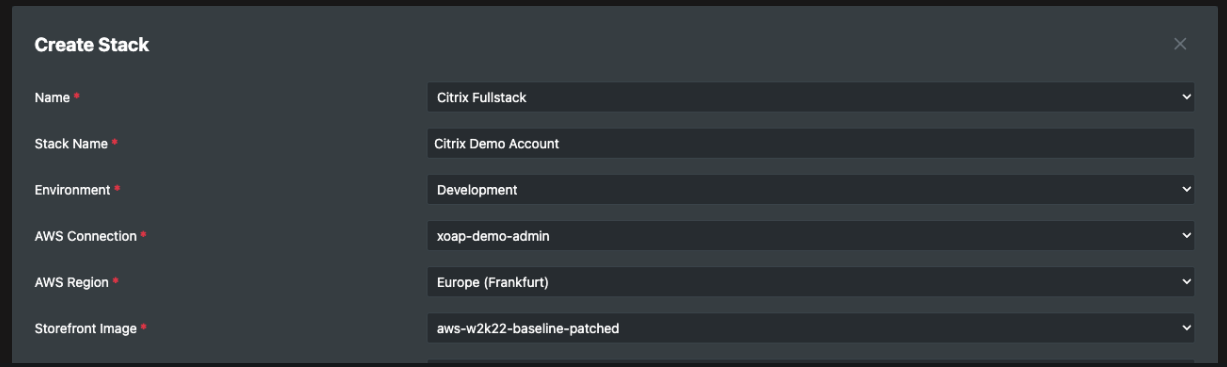
```
~/src/xoapv2/xoap-io/terragrunt-xoap-infrastructure/environment/core on dev !4 ?1 at 08:24:46
❯ terragrunt run-all plan
INFO[0000] The stack at /Users/mr_b/src/xoapv2/xoap-io/terragrunt-xoap-infrastructure/environment/core will be processed in the following order for command plan:
Group 1
- Module /Users/mr_b/src/xoapv2/xoap-io/terragrunt-xoap-infrastructure/environment/core/terraform-aws-stack-organizations

Group 2
- Module /Users/mr_b/src/xoapv2/xoap-io/terragrunt-xoap-infrastructure/environment/core/terraform-aws-stack-ecr
- Module /Users/mr_b/src/xoapv2/xoap-io/terragrunt-xoap-infrastructure/environment/core/terraform-aws-stack-email
- Module /Users/mr_b/src/xoapv2/xoap-io/terragrunt-xoap-infrastructure/environment/core/terraform-aws-stack-networking
- Module /Users/mr_b/src/xoapv2/xoap-io/terragrunt-xoap-infrastructure/environment/core/terraform-aws-stack-security

Group 3
- Module /Users/mr_b/src/xoapv2/xoap-io/terragrunt-xoap-infrastructure/environment/core/terraform-aws-stack-computing
```

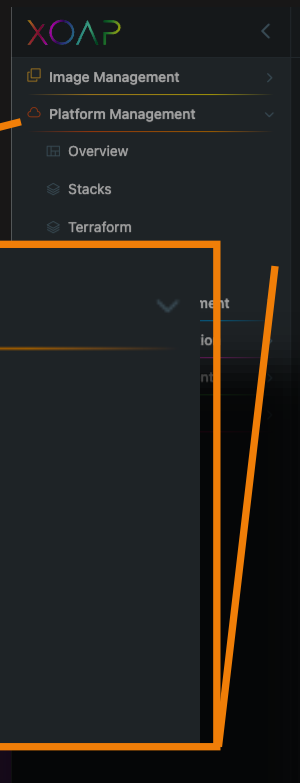
NEXT PART III

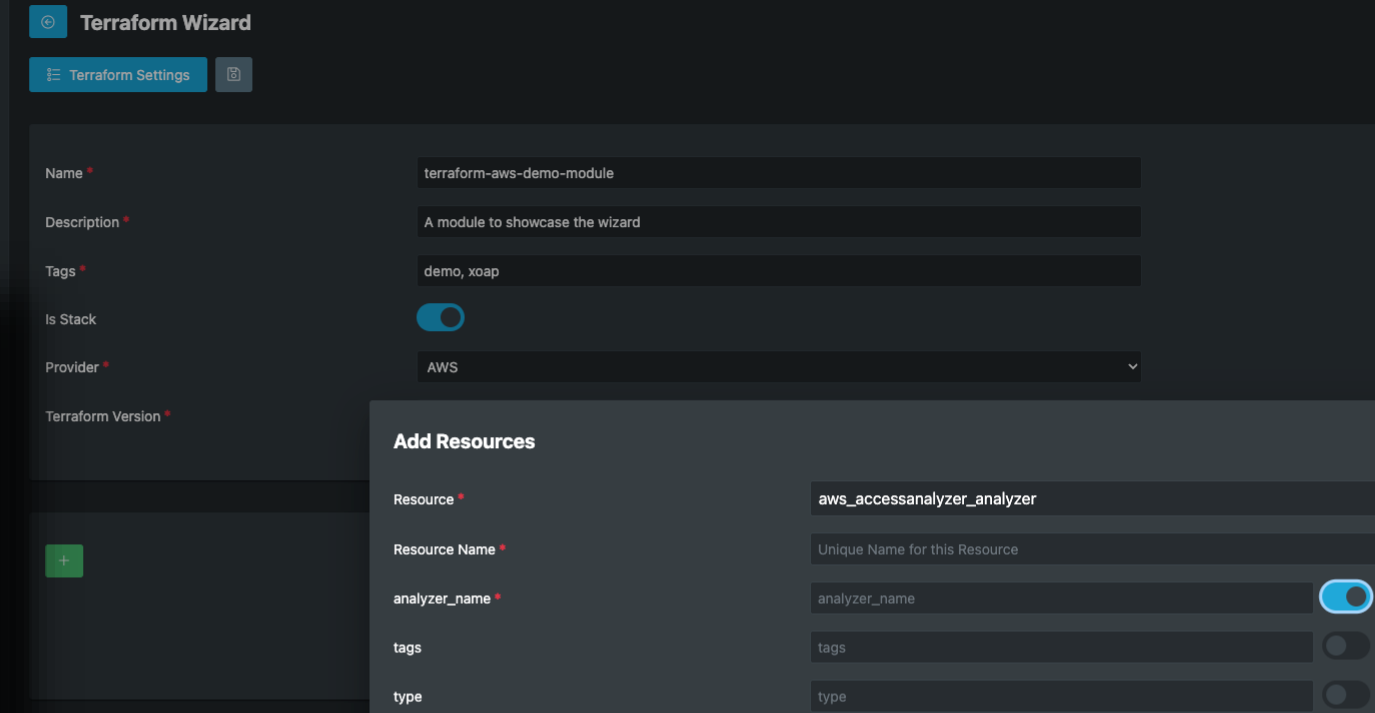
Building the Pipeline

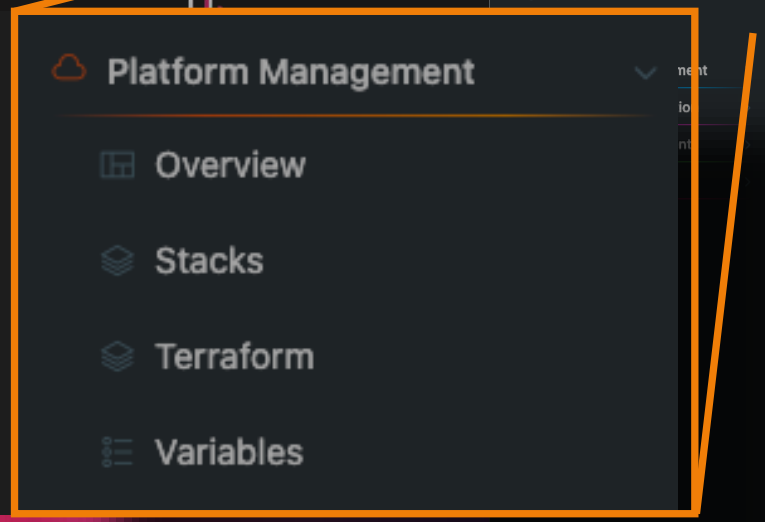
PLATFORM.XO

- Based on Terraform
- Stack Management
- Module Management
- Variables Management
- Resource Editing
- Platform Wizard
- XOAP Modules









Thank you for being here.

WE INVITE YOU.

XOAP
XTRA ORDINARY AUTOMATION PLATFORM

Want to give it a try?

Say…

hello@xoap.io

xoap_io

https://www.linkedin.com/company/xoap_io

https://github.com/xoap-io

© 2022 xoap.io all rights reserved