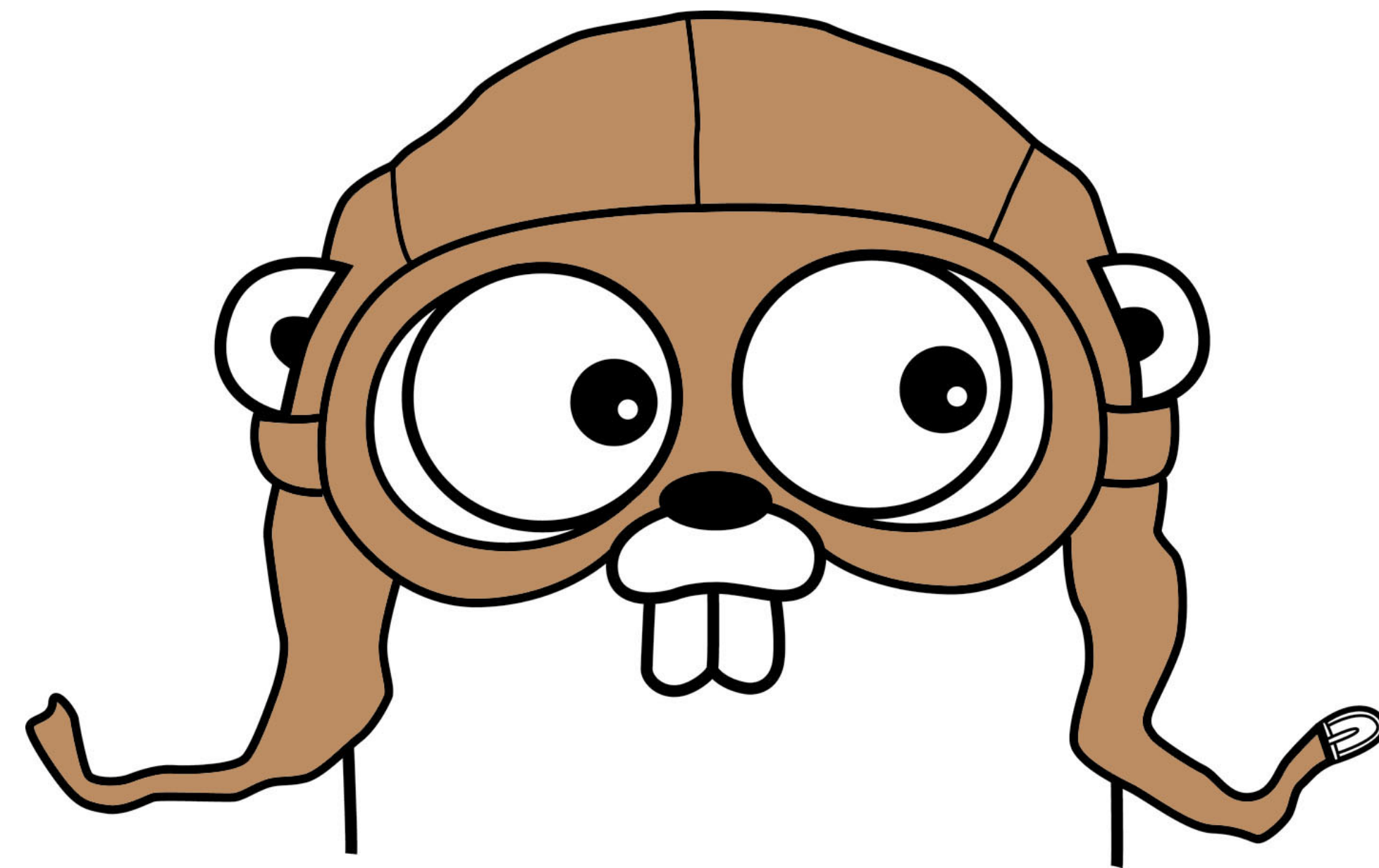


~~Building~~ (trying to build) a modern CI/CD pipeline for Micro-services

Short summary of our latest adventures in the world of CI/CD and GitOps

About me

- 🇧🇷
- 🎸
- + 10 years working with IT
- Backend Developer @ Ingrid
- Most asked question: Why Poland 🇵🇱?
Answer: Because I got a job here 😊



What is this talk not about?



- ArgoCD Installation
- GitHub Actions syntax
- Infrastructure setup
- Comparison between different CI/CD tools

Agenda

- What do we want to have?
- Pull vs Push model
- How does it look like?
- What does it change in our lives right now?
- What are the problems?
- Conclusion
- Questions



What do we want to have?

- GitOps approach 
- Continuous Deployment 
- No more deployments from CLI 
- More security 

Pull vs Push Model

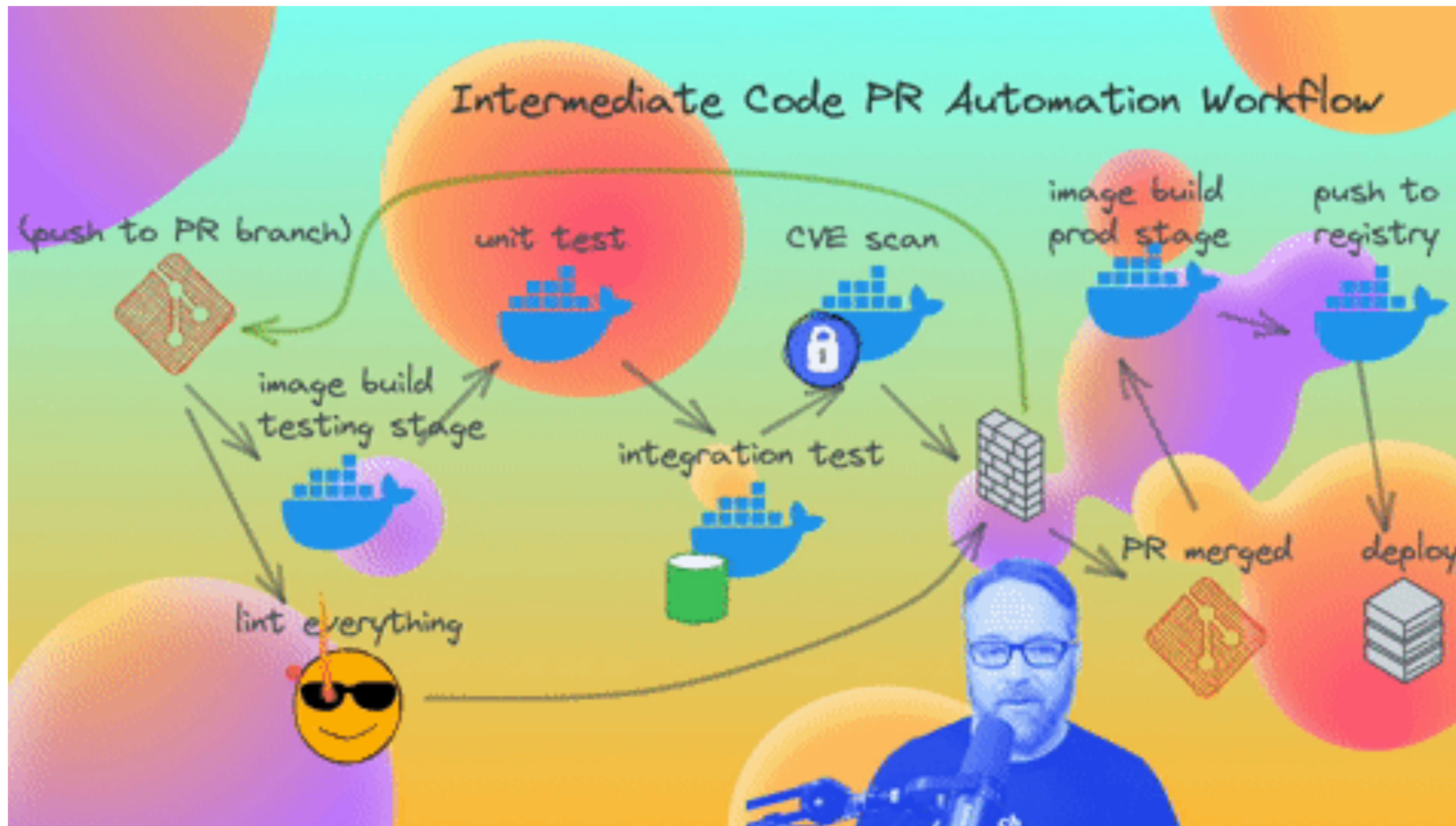
PULL

- Listen (pull) to Git repository for changes
- Secure - CD is deployed directly to the cluster
- Consistency - acts as an audit platform constantly checking the git vs environment
- Not a lot of tools on the market, the two big players here are ArgoCD and FluxCD
- Covers pretty much on k8s deployment.

PUSH

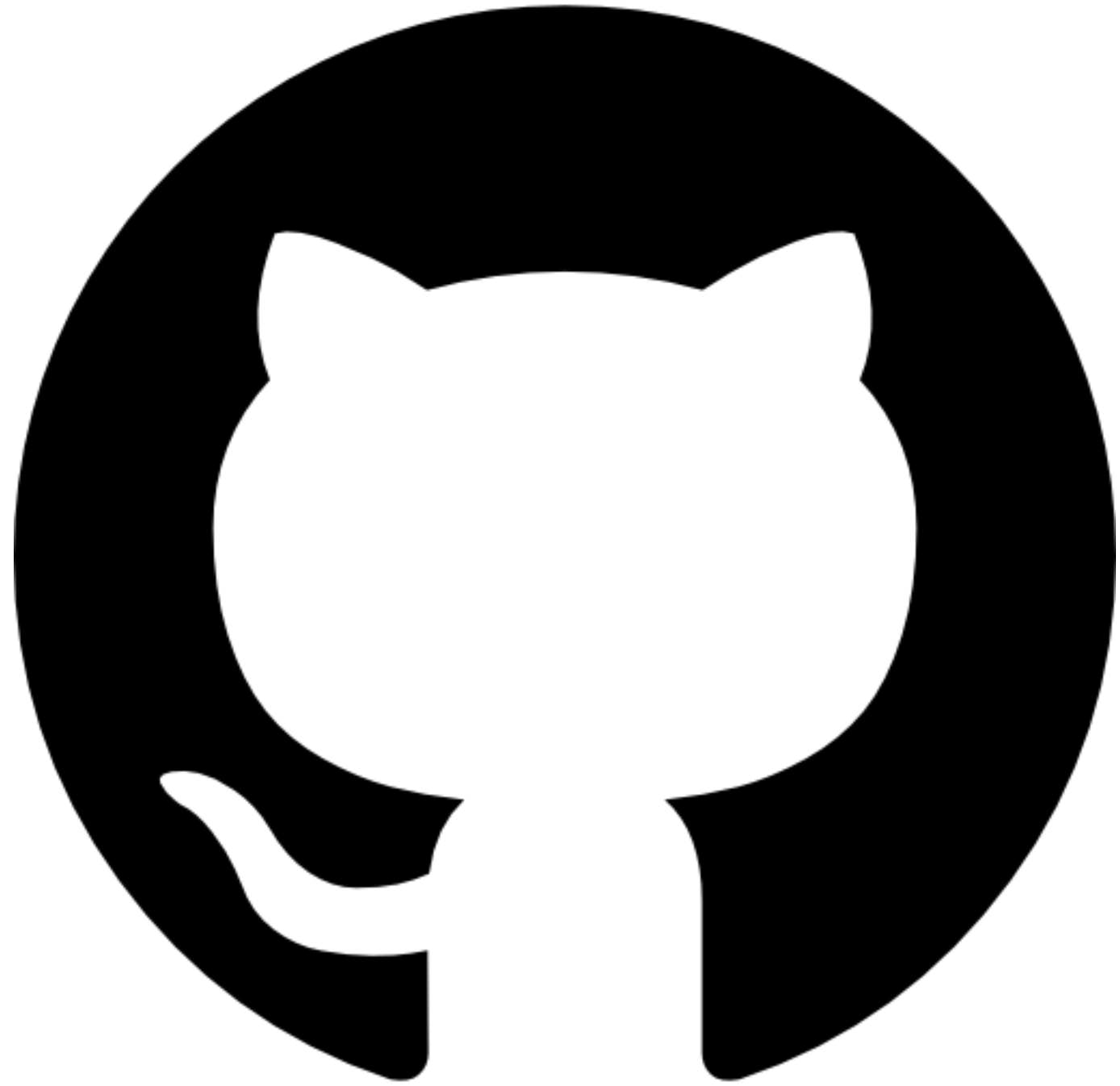
- Deployment is pushed from CI/CD tool (agent) to the environment
- Simple - it's basically all the CI/CD pipelines we've seen so far
- Flexible - more tools available
- You can deploy pretty much everything using the same tool (k8s, terraforms and etc)

Bret Fisher - Automation with Docker for CI/CD Workflows



How does it look like?





application code repository

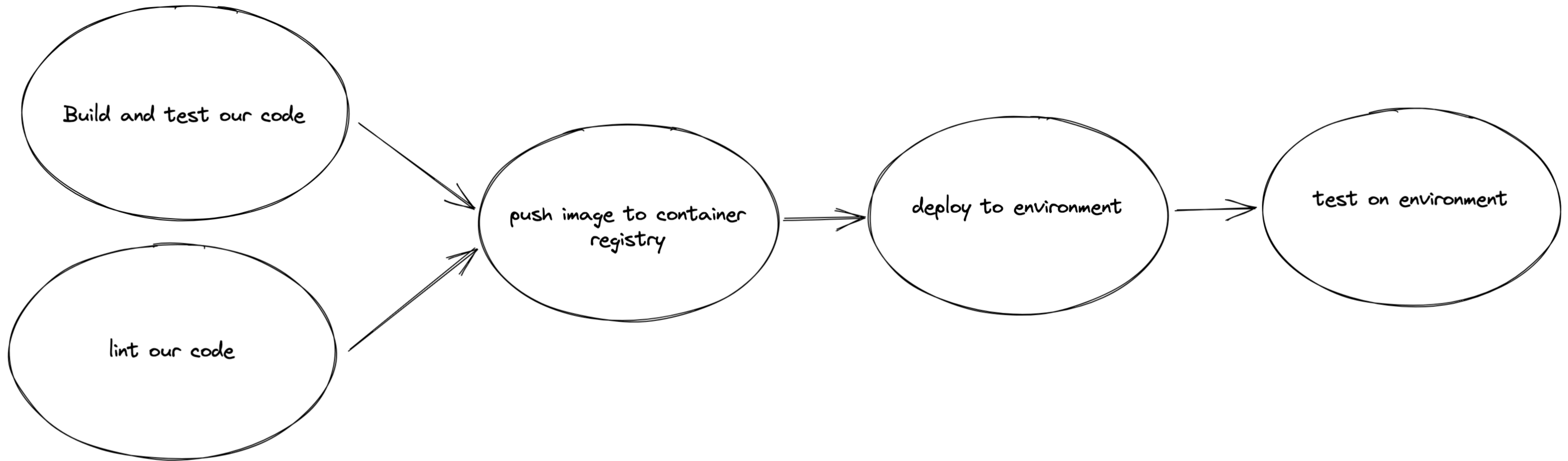
- the application code
- the workflow configuration for CI
- the workflow configuration for CICD
- secrets configuration on GH
- notification workflow

deployment code repository

- 4 branches:
 - common
 - development
 - stage
 - production
- k8s manifests for each app
- publish pre-release workflow

E2E tests code repository

- e2e tests to validate the deployment



What does it change in our lives now?

- In a matter of minutes your change will be deployed to dev, stage and with a PR ready to be merged (and deployed) to production
- We shouldn't introduce breaking changes. Everything should be compatible or behind a feature flag
- Re-running previous workflows from master causes re-deployment of older version
- You should take care of your production PR and make sure it gets merged and deployed

What are the problems?

- Syncing is hard, knowing when the application was deployed it's not trivial
- Concurrent changes are not easy to manage (multiple CI builds generating commits to the deployment repository), sometimes getting stuck on our wait for status step
- There are ways with GitHub Actions to create a more distributed pipeline but the maintainability seems harder and it's also harder to visualize the changes / deployments
- The patterns are not yet defined. You can easily find some quick setup that goes from build to deploy, but anything rather complex it's not an easy task
- Same is valid for the tools
- Observability is hard, you can track change via git log but it's not easy to answer how many times was this service deployed? Was there a re-deployment?

Conclusion

- Most of the problems you will face won't have an easy solution
- The GitHub Marketplace is really good and helps a lot
- It's great and easier to have your pipeline together with your source code
- To fully enjoy the benefits of it, the team needs to embrace the cultural change (not doing breaking changes, deploy more often and smaller chunks)
- It's not an easy road, you will need to get your hands dirty and develop / adapt some things
- In the end of the day, my personal feeling is that it's worth the trouble to try to implement it
- We didn't solve everything yet and probably with more usage of this new pipeline, more questions and challenges will appear

Presentation + Code



Questions ? ? ?

