Isn’t always good for you

Heather Osborn

Organic Isn’t Always Good for you

https://www.linkedin.com/in/heather-osborn/
Who am I?

Heather Osborn

25+ years in system engineering, devops and management.

- Private cloud/on prem
- Public cloud

Crazy cat lady

Distance runner

Immersive camping enthusiast
What’s the story?

- Creating an organic mess
- Background spaghetti
- Do you really know what’s wrong?
- Woah, that’s a big problem
- Yeet the whole thing
- Tools for fun and profit

Heather Osborn
Organic Isn’t Always Good for you
https://www.linkedin.com/in/heather-osborn/
STARTUP LIFE

What people think it’s like

What it’s actually like
Set up an application in a day!

Front-End Web & Mobile Development
Build and deploy secure, scalable mobile and web apps fast

Quickly launch web applications
Deploy scalable web applications in minutes without the complexity of provisioning and managing underlying infrastructure.

Websites
Reliable, highly scalable, and low cost website and web application hosting

Heather Osborn
Organic Isn't Always Good for you
https://www.linkedin.com/in/heather-osborn/
Background - Organic Spaghetti is especially bad for you.
Let’s figure out what the problems are…
<table>
<thead>
<tr>
<th>Team 1</th>
<th>Team 2</th>
<th>Mobile</th>
<th>Backend</th>
<th>Team 3</th>
<th>Team 4</th>
<th>Data</th>
<th>Security</th>
<th>DevOps</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODE has old monolith, calendaring, not microservice</td>
<td>ODE tests server code separately, then needs to test microservices in the release environment.</td>
<td>Soyk doesn't fail pipelines for security vulnerabilities (allows tech debt to accrue)</td>
<td>Waiting on Cypress jobs</td>
<td>Too many unused feature flags - recommend to remove when done</td>
<td>Critical mass of total tests</td>
<td>Data is a full distinct platform from app platform</td>
<td>Account configuration doesn't allow cross-account AWS keys</td>
<td>Inconsistent processes (deploy to dev means different things to different teams)</td>
<td>Multiple ways to configure local environment</td>
</tr>
<tr>
<td>Environment variables are difficult to track (some in one app, but not consistent)</td>
<td>Time to seed db - always needed with data migration</td>
<td>Microservices have multiple developers contributing so release doesn't stay consistent</td>
<td>Cutting a release is manual, on-call person kicks it off on Friday via GHA</td>
<td>Working with microservices is complex, need to provide PAM understand list, right version of client app</td>
<td>Unit testing for microservices</td>
<td>Tech debt - no follow-up (versions of product).</td>
<td>Inconsistent term/abstractions (managed in different repos, different per environment - many duplicates and some manually configured)</td>
<td>Consistency</td>
<td>Consistency</td>
</tr>
<tr>
<td>Server build times</td>
<td>Changes in main and beta, dependencies are not merged, no release experience drift</td>
<td>Microservices tags are changed with every merge to main, requiring re-verification</td>
<td>Testing microservice dev with ODE needs to have the same policies with list/COBS</td>
<td>Different runtime/build environments - differing versions of react, etc.</td>
<td>Link rules are inconsistent</td>
<td>Queue is manually managed - can't automatically pause when there's an issue</td>
<td>AWS best practice to not follow (version checking in pipelines)</td>
<td>Lack of consistent environment to test</td>
<td>Environment Test</td>
</tr>
<tr>
<td>Documentation of commonly encountered problems ( seeding and ODE troubleshooting)</td>
<td>Microservices tags are changed with every merge to main, requiring re-verification</td>
<td>Unable to determine change failure rate because e2e testing is currently broken</td>
<td>Microservices base template - come conventions make it difficult to track - tagging uses 2nd to last GHA and you need to manually compare. Script could be updated to tag properly</td>
<td>No way to manually test the onboarding flow (can't inject pathway through to troublesome specific config files)</td>
<td>Feature flags - need to sync with prod</td>
<td>Seeding - what are we seeding, how frequently, when it is reset</td>
<td>Can spin up client/server locally but not microservices</td>
<td>Lack of consistent environment to test</td>
<td>Complexity</td>
</tr>
<tr>
<td>Server build times</td>
<td>Local testing usually used for e2e</td>
<td>Release environment adds days to cycle which causes more drift</td>
<td>MTTR is currently 48 hours. Deployment is using a developer mac to GHA, but requires appstore deploy and review</td>
<td>If deploy fails using helm, it will try indefinitely (out of sync error)</td>
<td>Product manager and design use release because feature flags are closer to production (otherwise you need to recycle from scratch)</td>
<td>Feature flags - need to sync with prod</td>
<td>Database upgrades take out data ecosystem</td>
<td>No consistent source of truth for config and keys</td>
<td>Complexity</td>
</tr>
<tr>
<td>Local testing usually used for e2e</td>
<td>Competition to use release which makes ODEs more appealing. Most people test locally unless it's sensitive</td>
<td>Ideal would be flexible ODE where you can choose your dependencies</td>
<td>Native deployment too frequent (refreshes version of app which code push doesn't)</td>
<td>Retain logs for longer (eng should check logs immediately after release)</td>
<td>Team ownership of RDS updates</td>
<td>Need testing parity in the app ecosystem</td>
<td>No tagging of resources</td>
<td>Unlabeled or mislabeled infrastructure</td>
<td></td>
</tr>
<tr>
<td>Microservices dev is cherry pick from beta branch</td>
<td>Microservices dev is cherry pick from beta branch</td>
<td>Takes a while to stand up release</td>
<td>No standard for local testing</td>
<td>No stack alerts for microservice deploy failures on prod</td>
<td>No documentation/infrastructure books - want ODE with monolith/microservices</td>
<td>Want ODE with monolith/microservices</td>
<td>Engineer access of production resources needs process for manager approval and time limits</td>
<td>Out of date documentation</td>
<td></td>
</tr>
<tr>
<td>Data on release is better, ODE and PP aren't useful/current</td>
<td>Data on release is better, ODE and PP aren't useful/current</td>
<td>Release would be better with data closer to master</td>
<td>Dependencies on Auth for microservices - where you need it changes how you access it (local, release, etc)</td>
<td>Dependencies on Auth for microservices - where you need it changes how you access it (local, release, etc)</td>
<td>Lack of runbooks for failover backups</td>
<td>Complicated networking prevents new software from being implemented</td>
<td>Lack of runbooks for failover backups</td>
<td>No cohesive documentation for tracking external communication</td>
<td></td>
</tr>
</tbody>
</table>

Heather Osborn
Organic Isn't Always Good for you
https://www.linkedin.com/in/heather-osborn/
What was DevOps going to do?

Choice 1: Greenfield?

Choice 2: Upgrade/clean up prod, copy to dev for parity, update prod, repeat

Choice 3: Improve in place
Greenfield Go Ahead
What do we want?

- Never ever do anything that’s not in code
- Validate validate validate
- Keep documentation close to code
- Simplify product engineer life
- Testability
- Consistent environments
- Low touch release process with visibility
  - Consistent promotion process
- NO SPAGHETTI
It’s so clean…

Platform cluster (tools)
- Alpha
- Beta
- Stable

App cluster
- Alpha
- Beta
- Stable
Visualize Consumption of Application Templates

Key

- Github Repo
- Information Bubble
- Helm Artifact
- Docker Image

Inheritance
Pull
Push (Deploy)
Potential

GitHub
platform-stable
ArgoCD

auth-service
Argo Application
- Helm Values
- Docker Image (worker)
- Helm Chart
- Set Namespace
- Argo pulls worker stored Helm Charts

polaris-gateway
polaris-harbor
polaris/twinegate

future-microservice
server
aut-service

application-next-template
application-go-template
application-base-template
terraform-application-base-template

Chain of Thought Action for application-base-template
- Create Docker Image
- Push to Harbor

Base for Specific Languages

https://www.linkedin.com/in/heather-osborn/
Validate EVERYTHING

Validate in local - pre-commit hooks
Validate in CI - conftest
Deployment validation - Gatekeeper/OPA
Validate cluster runtime - GuardDuty
Any Questions?