DevOps Essentials DevOps Culture Notes

The Goals of DevOps

DevOps Culture

* DevOps Culture is about collaboration between Dev and Ops
* The traditional separation between Dev and Ops makes Dev and Ops have different and opposing goals
  + Dev wants speed
  + Ops wants stability

One of the worst things for stability is changes this makes the goals of both collide. Each small change brings up a risk to stability.

DevOps culture is about collaboration between Dev and Ops. They both work together to achieve the same goal of speed and stability.

The Goals of DevOps Culture

* Dev and Ops are playing on the same team
* Dev and Ops have shared goals and shared measurements

These goals include

* Fast time-to-market. The time code being finished and deployed to production and working properly.
* Few production failures. This goal is based around stability. Find solutions that allow us to get that fast time-to-market while minimizing the number of failures that are caused by those changes.
* Immediate recovery from failures. Recovering from failures as quickly as possible even sometimes immediately if you include automation.

A Story of DevOps vs. Traditional Silos

Traditional

Separated Developer, QA, and Operations teams

1. Devs write code
2. Throw it over the wall to QA. (Developers are no longer responsible for that code until the QA team bounces it back to them)
3. QA finds a problem and sends the code back to development team. Bounces back until QA deems the code ready.
4. If QA team deems it ready, they throw it to operations. Operations can throw it back to dev if they find a problem.
5. Each groups domain is a “black box” to the other groups.
6. Problems lead to finger pointing with both groups claiming it is not their problem.

What went wrong?

* Devs don’t trust the operation teams and have no insight on how production works.
* Ops don’t trust the Dev team and have no insight into the code.
* This creates a black box scenario where all they can do is point fingers. Our code works on our systems says the Dev team, while the operation teams states it caused a problem in production so fix it. It becomes a stand still scenario.
* They also have different priorities which pins them against each other

Even if both teams want to work together the managers are holding each team accountable for their specific goals.

Dev team manager needs them to deliver features fast.

Ops team manager needs production to be stable.

Downsides of Traditional

* “Black boxes” lead to finger pointing
* Lengthy process means slow time-to-market
* Lack of automation means things like builds and deployments are inconsistent. Deployments may be deployed differently depending on who is deploying it.
* It takes a long time to identify and fix problems.

The Story of Some Code in a DevOps Model

1. Devs write code
2. Code commit triggers automated build, integration, and tests
3. Because steps are automated QA gets their hands-on code much quicker. Allows for constant testing.
4. Once it is ready, kicks off an automated deployment to production.

These steps required collaboration between all these teams

Because steps are automated, it is much easier to deploy code in a stable fashion

This allows deployments to occur much more frequently, getting features into the hands of customers faster

What happens if a deployment breaks something in a DevOps environment?

One of the most important features of DevOps is automated monitoring that detects breaks and problems immediately and notifies the team. The team is then able to use the automated system to rollback and deploy a previous working version of the code to fix the problem almost immediately.

What went right?

Dev and Ops worked together to build a robust way of changing code quickly and reliably.

* Both Dev and Ops worked together to prioritize both speed of delivery and stability

Automation led to consistency

* Building, testing, and deploying happened the same way every time
* It also happened much more quickly and much more often

Good monitoring

* Dev and Ops worked together up front to build good processes
* Even though a code change caused a problem, users experienced little or no downtime

Why do DevOps?

Happier Teams

* Tech employees tend to be happier doing DevOps than under traditional silos
* More time innovation and less time putting out fires
* Devs don’t feel like they have to fight to get their work out there
* Operations people don’t have to fight Dev to keep the system stable

Happier Customers

* DevOps lets you give customers the features they want quickly
* And you don’t have to sacrifice stability to do it