How E-Verify Deploys Over 300 Times a Month

By: Dan Davis

Excella
excella.com | @excellaco

Helpdesk Ticket

Hi E-Verify Team,

I’m on the phone with a user that is having trouble with the registration page. He said …[details about the bug…]

What do you want me to tell them to do about it?

Thanks,

Help Desk

Helpdesk Ticket

Hi Helpdesk!

We noticed that problem earlier today and we have a fix that should be deployed in about 15 minutes.

Could you tell the user to just go grab some coffee or something?

Thanks,

E-Verify Team

Don’t let process interfere with work!

- No discussion of versions / release windows
  - Our team is constantly releasing software
- No coordination with the QA team about regression testing
  - Our tests are automated and we trust them
- Already knew about the problem
  - We don’t rely on users to tell us about problems

This is E-Verify…this is why we can deploy over 300 times a month...

Why our team is successful

Extensive Automation

Automated Testing

Monitoring / Alerting

Extensive Automation

Deployments are a routine activity

Automation

Automate Everything

- Infrastructure as code
  - We rebuild our infrastructure as needed
  - Don’t nurse servers back to health
- Scalable
- Auditable
  - Security
  - Environment consistency (configuration)

Automation

Each service has its own CI/CD Pipeline

Pipelines are responsible for

- Creating build artifacts (container images)
- Quality Checks (internal and external)
- Provisioning Necessary Infrastructure
- Deploying to Environments

All of this is maintained in code (pipelines as code)

Automation

- Pipelines must be ***fast***
- Fail quickly
  - Run tests as soon as possible
  - Keep developers engaged
- In production quickly
  - This is true agility…we can react!
  - Don’t let slow tests bog the pipeline down
  - Typically takes about 30 minutes

Trust Your Tests

How We Test

Find the right **level** of testing

- Focus heavily on unit/integration tests
- Acceptance tests should cover only core business functionality
- Try to minimize complex, end-to-end testing
  - Brittle, slow, unreliable

How We Test

Build independently testable microservices

- API test is faster than browser testing
- Well-defined service with discrete functionality
- Easier to setup test data (fixtures)

Use health/dependency checks

- Service track if they can hit dependencies

Trust in your test suite

Tests should prevent

- Catastrophic failure
- Major bugs

Don’t need to have an exhaustive test strategy

- Prevention of **all** minor bugs is not economical
- We can introduce changes quickly

Monitoring / Alerting

We’re always watching you…(but not in a creepy way)

Monitoring / Alerting

- What happens **after** we deploy?
  - Flood of users hit our systems
  - Memory leak could sap performance
  - Network failures

- We use monitoring tools to gain insight into our live applications
  - Setup alerts to know when to pay attention

Alerting

- Alerts should
  - Only be triggered when we need to do something
  - Be displayed in places where developers can see them
  - Provide actionable information
  - Notify us when they are resolved

- Our team should be **proactive**
  - Don’t wait for users to tell us the system is broken

Why our team is successful

Extensive Automation

Automated Testing

Monitoring / Alerting

Questions?

www.linkedin.com/in/daniel-davis-56310b3a