DevOps at Mingle Analytics

Story and Lessons Learned

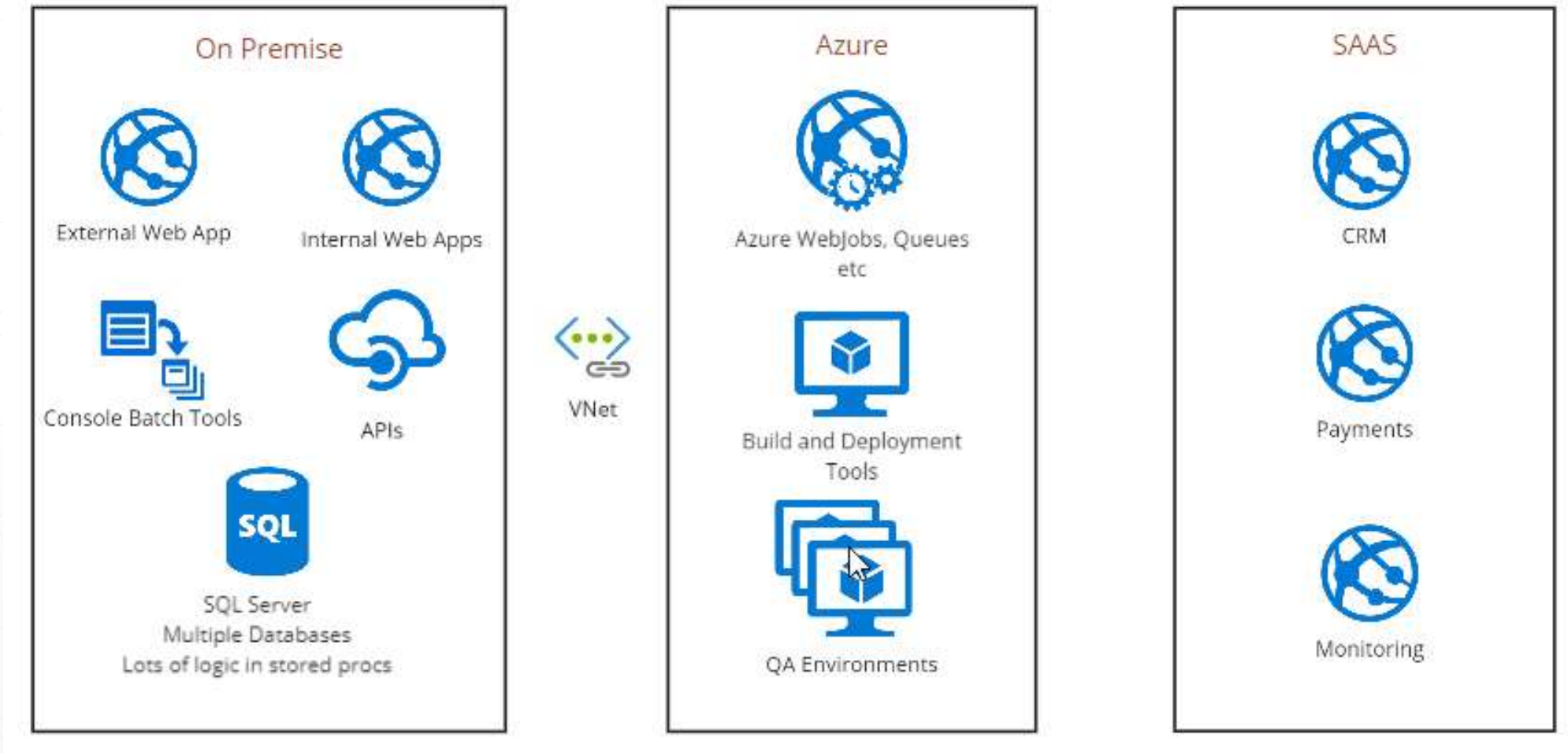
Floyd Hilton - Tacoma Software

- Twitter - @fhilton
- LinkedIn - https://www.linkedin.com/in/floydhilton/
- Email - floyd@tacomasoftware.com
- Blog - http://floydhilton.com/

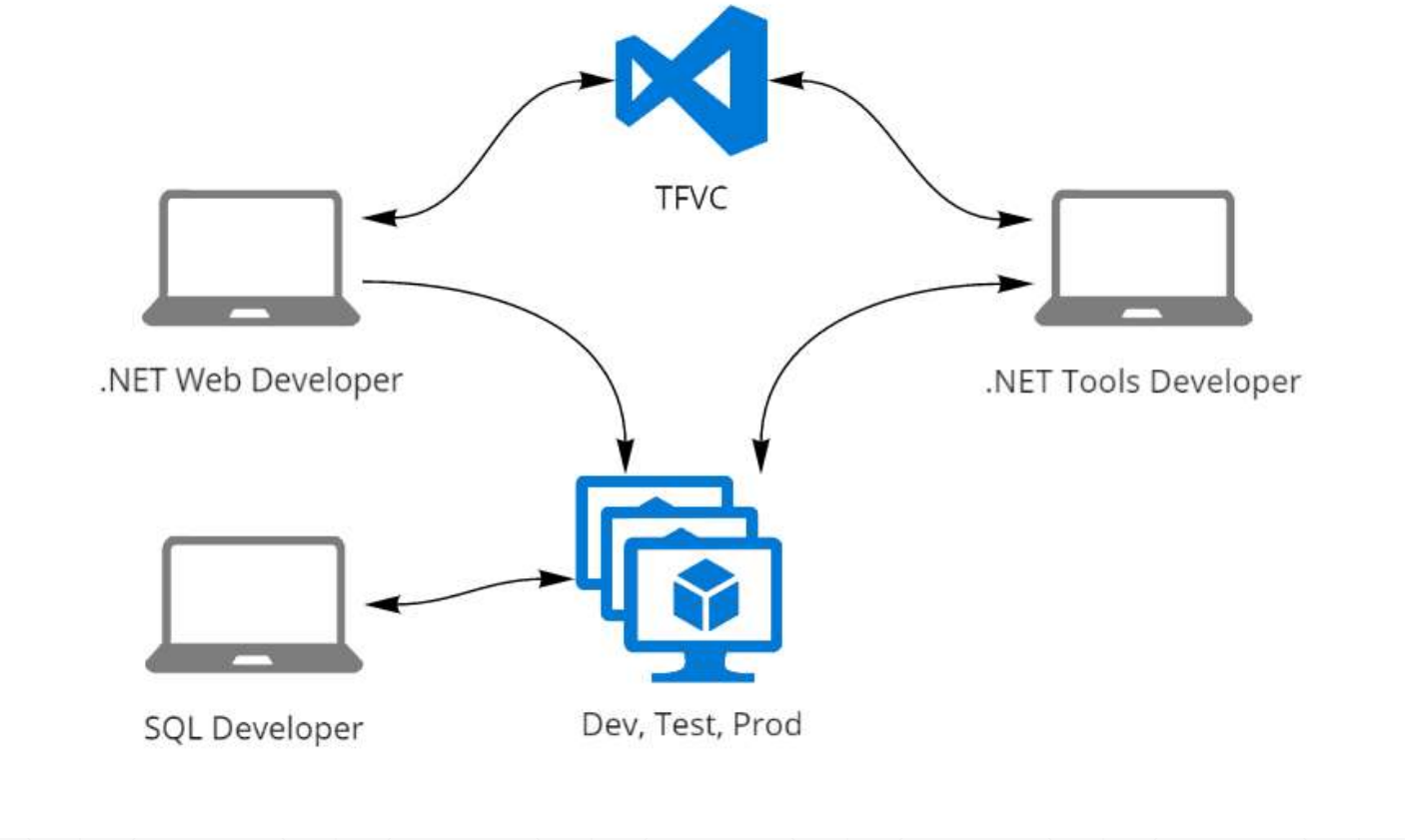
Background

- Mingle Analytics is an industry leader in Medicare reporting
- Worked with Mingle from 2014 to 2017
- Brought in to help accelerate legacy re-write
- Moved from no automation to automated integration and deployment

Architecture Overview



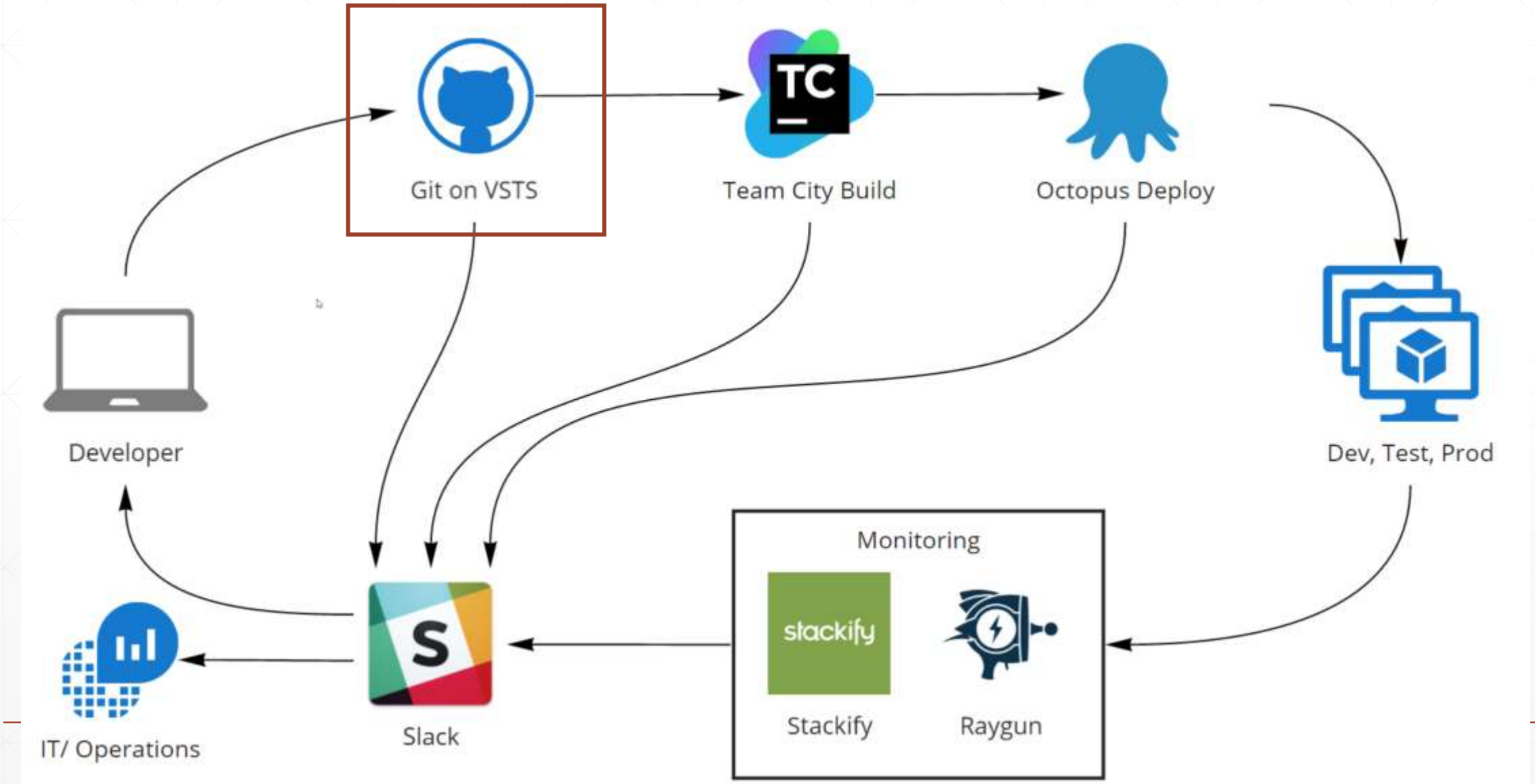
In the Beginning



Have a Reason for DevOps - Problems to Solve

- Painful merging, lost changes
- Quality issues
- Long regression test cycle = opportunity cost
- Difficulty tracking down problems
- Painful releases dependent on one key person
- Duplicated efforts and confusion between teams
- Unable to keep up with customer needs

Integration, Deployment and Monitoring Overview



Source Control

Move from TFVC to Git

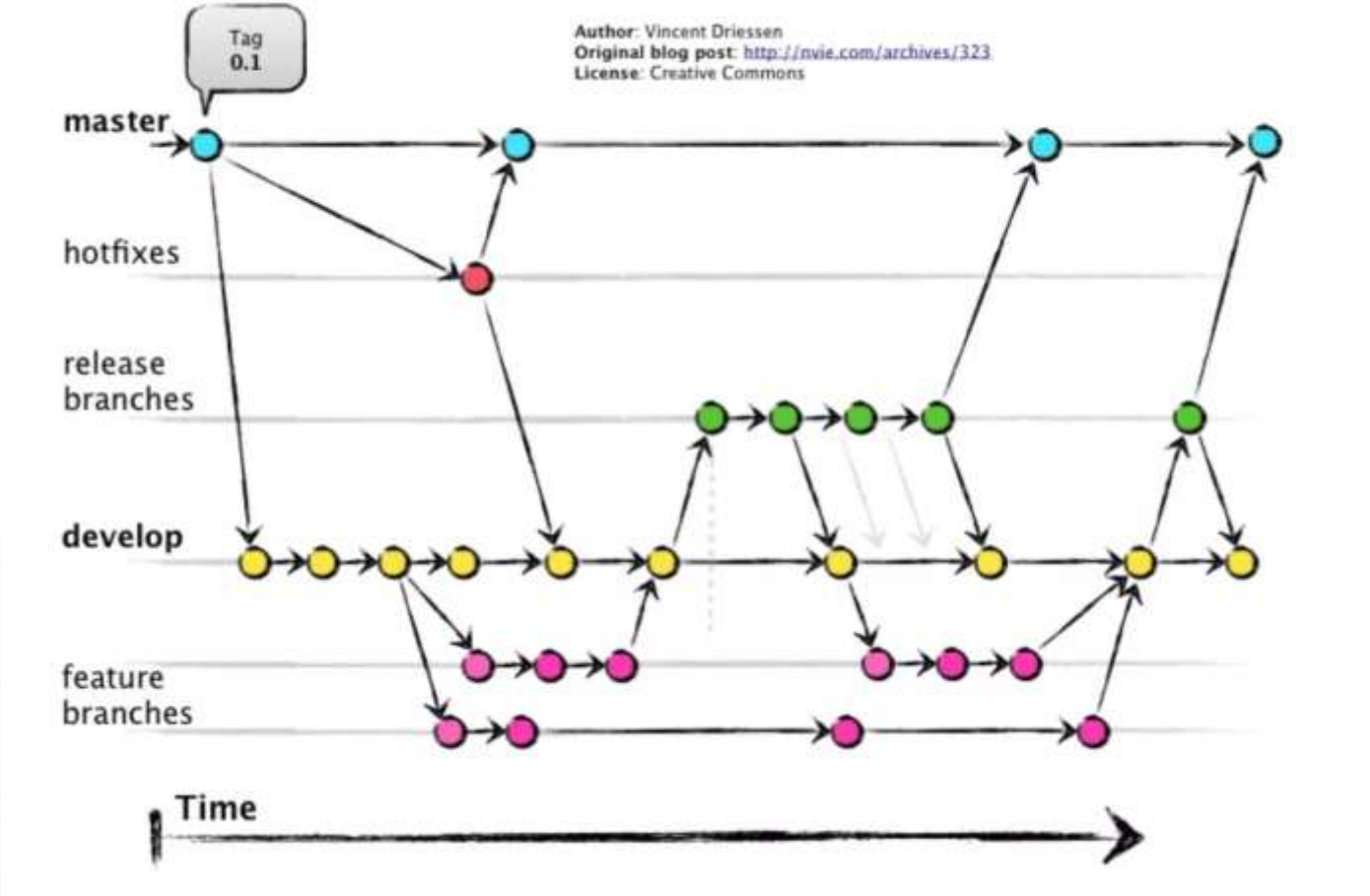
Benefits

- Don’t need VPN for source control = more frequent commits = less merging issues
- Quick and easy branching
- Documentation in Git using Markdown

Learnings

- Training on Git saves time later
  - Understand how it works
  - Remotes
  - Branching
  - Merging and Rebasing
  - Cherry picking
  - Pull requests

Implement Git Flow Process



Implement Git Flow

Benefits

- Know what code is in production helps with fixing issues
- Hot fixes are as simple as a branch and a pull request
- Allows coding on new features to continue while a potential release is in QA

Learnings

- Need to account for drift between production and the master branch (especially the database)
- Need to be diligent with merging changes between master, release, and develop branches
- If possible use feature flags and have one master branch

SQL in Source Control

Benefits

- Found many broken views and stored procedures
- Facilitated SQL unit testing and deployment
- Quickly track down errors by knowing what changed and when
- Added transparency between teams
  - All teams doing pull requests

Learnings

- Need to account for drift
- Importing and fixing takes time
- Build times
- State of test vs production can cause unexpected issues
- Training and acceptance by all
  - Can use even if everyone else does not

Git Tools and Resources

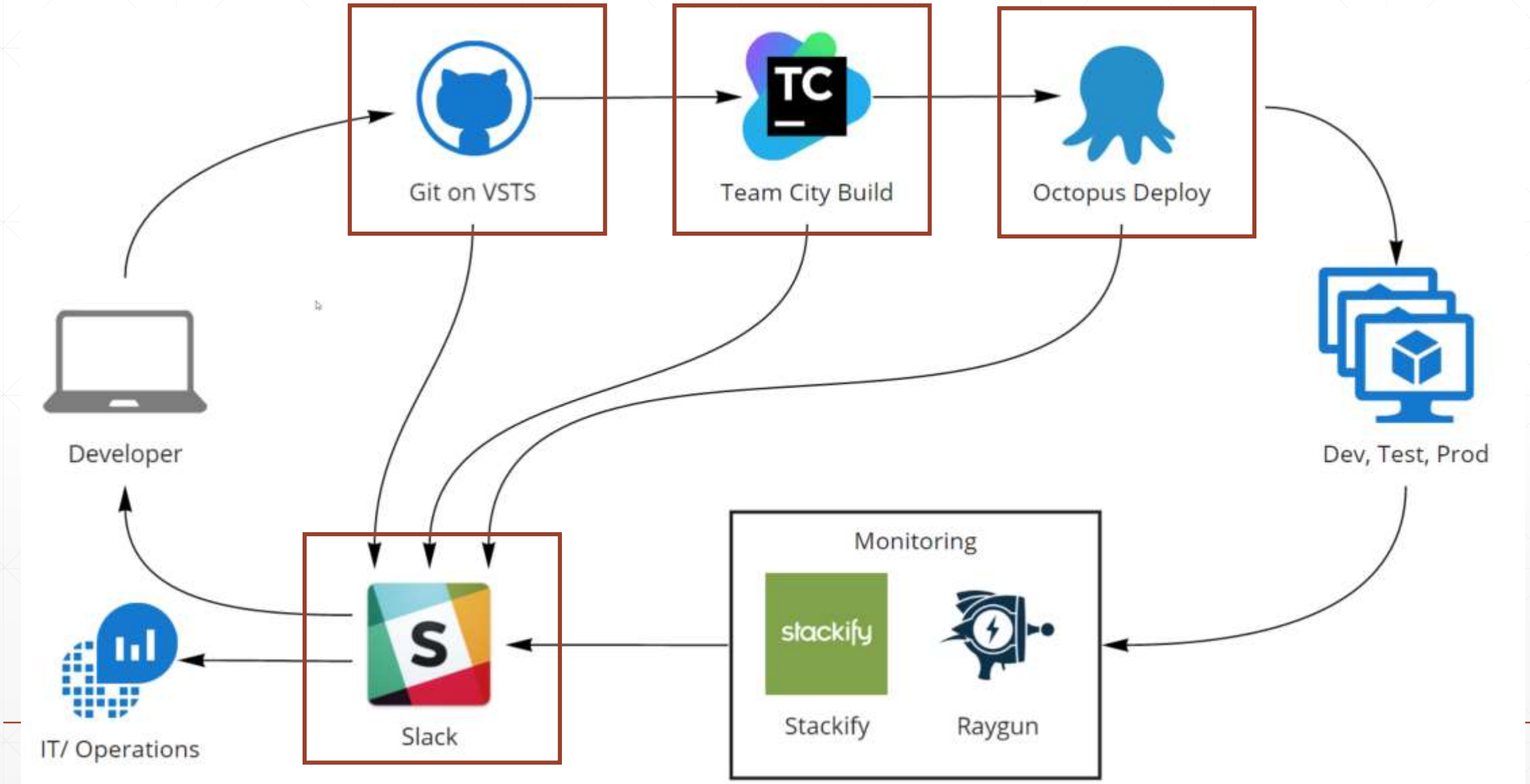
- Sourcetree - https://www.sourcetreeapp.com/
- Beyond Compare - https://www.scootersoftware.com/
- Tutorials - https://www.atlassian.com/git/tutorials
- Git Flow - https://datasift.github.io/gitflow/IntroducingGitFlow.html
- Git Hosting
  - GitHub - https://github.com/
  - Team Services - https://www.visualstudio.com/team-services/

SQL in Source Control Tools and Resources

- Database tools video: https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/SQL-Server-Data-Tools-for-Visual-Studio
- Red Gate ReadyRoll: https://www.red-gate.com/readyroll/
- DbUp: https://dbup.github.io/

Continuous Integration

Integration, Deployment and Monitoring Overview



Implement Continuous Integration

Benefits

- Eliminated “Works on my machine”
- Found issues sooner by running tests with every push
- Tagging of builds helped find problems

Learnings

- CI server updates (Use SAAS if you can)
- Broken builds can take up a lot of time, have a process in place that notifies the team when a build breaks and who broke it
- Building pull requests prior to merge can help a lot with broken builds
- Team City limits on # of build servers

Implementing Pull Requests

- Using Visual Studio Team Services – Very similar to Git Pull Requests
- Gated
  - Related to a user story
  - Approved by one other developer
  - Must have unit tests
  - Code must build, deploy, and tests must pass
- PR notifications go to Slack with @mention
  - New PR, acceptance, rejection, build and test failures

Implementing Pull Requests

Benefits

- Drastically lowered the number of broken builds
- Improved test coverage
- Code review on every PR improved code quality and consistency

Learnings

- Easy for PRs to get held up waiting for other developers to review
- Need to manage how “Picky” the code reviews are
- Team City build server limitation slowed process on busy days

CI Tools and Resources

- Team Services CI - https://www.visualstudio.com/team-services/continuous-integration/
- Donovan Brown - http://donovanbrown.com/
- TeamCity - https://www.jetbrains.com/teamcity/

Testing

Implementing Unit and Integration Tests

Benefits

- Find bugs before they are merged and released
- Cleaner, less coupled code

Learnings

- Some code bases are harder to unit test than others
- Writing tests is a skill that takes time to develop - training
- Enforce unit tests using pull requests

Implementing User Interface Tests

- Used a combination of Selenium and SpecFlow (similar to Cucumber)
- Developers created a custom test framework for the applications and then QA used Gherkin syntax with SpecFlow to create tests.
- Tests could be run from the CI server but were limited due to available environments

Implementing User Interface Tests

Benefits

- Lowered regression test time by automating basic tests
- Allowed synthetic monitoring of applications in production

Learnings

- Tests are brittle and difficult to create and maintain
- Would not bother with SpecFlow or Gherkin if doing again
- Only use for smoke testing or very basic regression
- May be the option for legacy applications that are not written to be tested

Combined Development and QA

- Involved developers in regression testing
  - Used Visual Studio Team Services to script and run manual regression tests
  - Easily see testing progress, track issues, and link stories
- Combined QA and Dev groups
  - Aligned goals
  - Brought transparency
  - Reduced duplication of effort

Testing Ratios, Unit vs Integration and UI Visual Studio Team Services at Microsoft

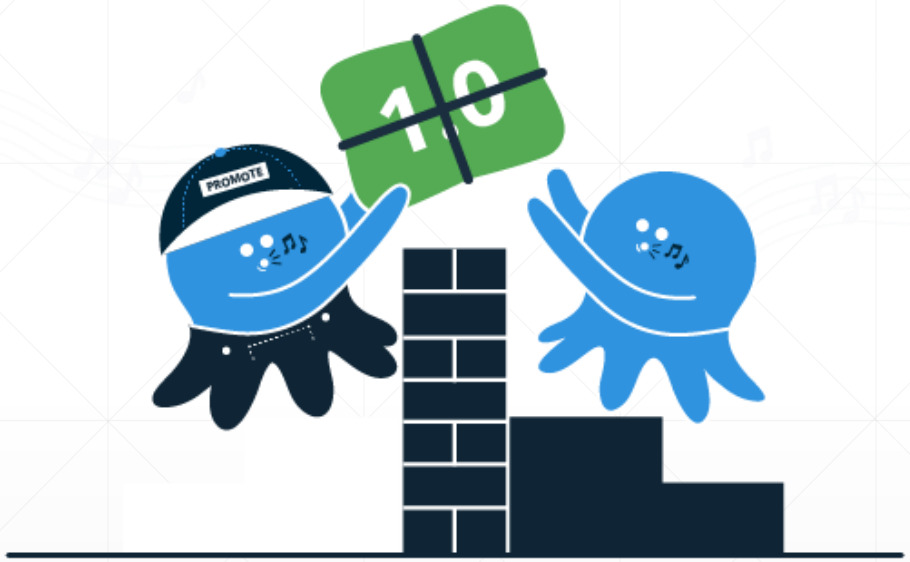
Testing Tools and Resources

- Uncle Bob - http://blog.cleancoder.com/
- .NET Rocks, Brian Harry - https://dotnetrocks.com/?show=1496
- Video - Slacker with Database Project and Team Services
  - Part 1 - https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/SQL-Server-Data-Tools-for-Visual-Studio
  - Part 2 - https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/SQL-Server-Data-Tools-in-your-DevOps-pipeline
  - Part 3 - https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/SQL-Server-Database-Unit-Testing-in-your-DevOps-pipeline
- tSQLt - http://tsqlt.org/
- Slacker - https://github.com/vassilvk/slacker/wiki/Slacker-Tests

Deployment

Automated Deployments

- Started with TFS using msbuild, very limiting
- Moved to Octopus Deploy
- Automated deployments done for every pull request
- Deployed on Prem and in Azure
  - Multiple IIS Websites and APIs
  - Console Apps
  - Azure Queues and Webjobs
  - Multiple SQL databases
- Allowed configuration of variables by environment
- Notifications of success and failure to Slack



Automated Deployments

Benefits

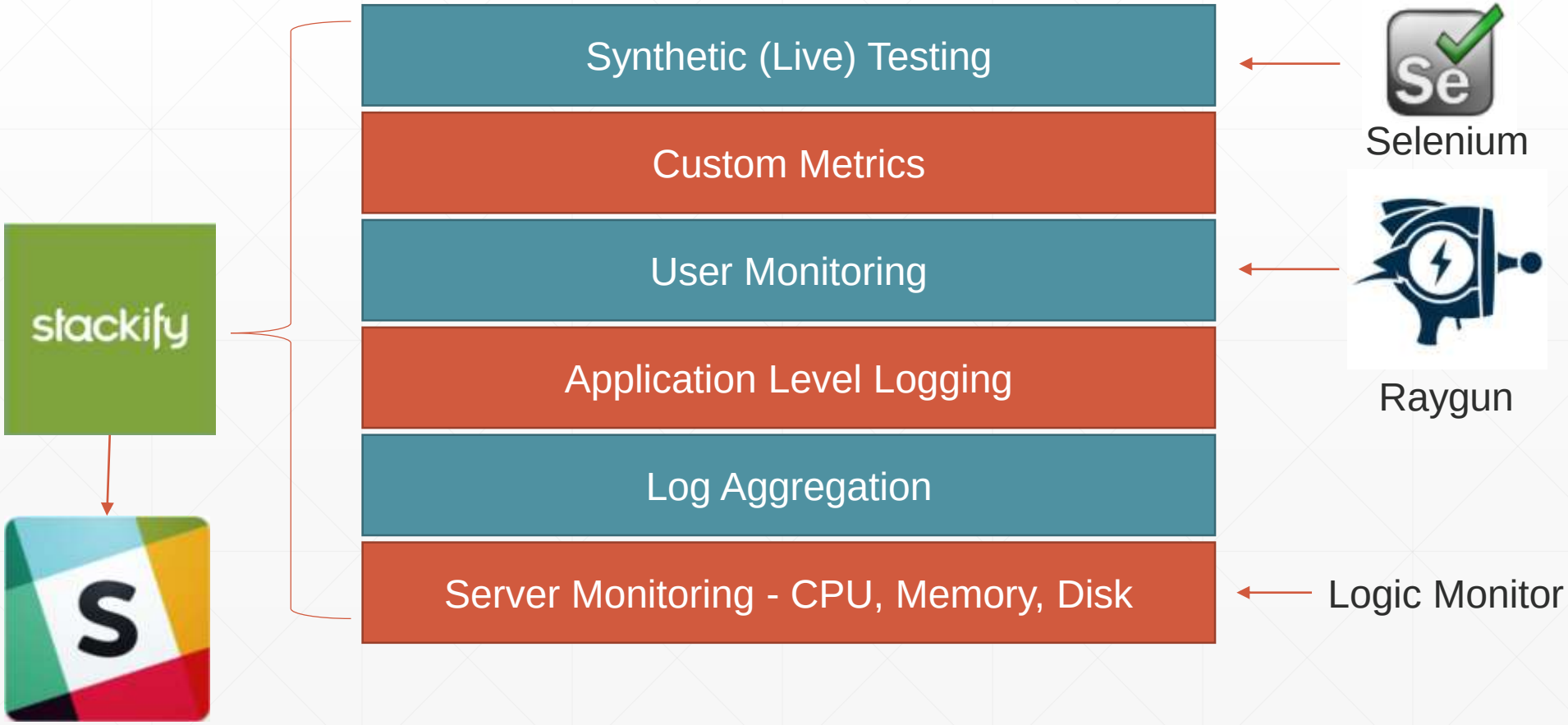
- Decreased release times
- Deployments on every PR means less surprises when going live
- Scripting of IIS and other settings for more consistent environments
- Easy deployment to multiple environments
- Dev not required for deployment

Learnings

- Variables can get complex fast, think about ahead of time
- If applications are coupled, test and deploy them at the same time
- Non SAAS, no auto update, need to keep tool up to date

Monitoring

Monitoring


Synthetic (Live) Testing
Custom Metrics
User Monitoring
Application Level Logging
Log Aggregation
Server Monitoring - CPU, Memory, Disk
Selenium
Raygun
Logic Monitor
stackify

Monitoring

Benefits

- Wholistic view of all systems
- Much easier to find problems
- Pinpoint performance issues between application and SQL server
- Set thresholds and receive alerts
- Brought together Dev and Ops

Learnings

- Make sure you have access to your data
- Use Log Levels – Turn on when needed
- Alerts should be real and infrequent – noise is ignored
- Log feature use – 50% features not used by most users

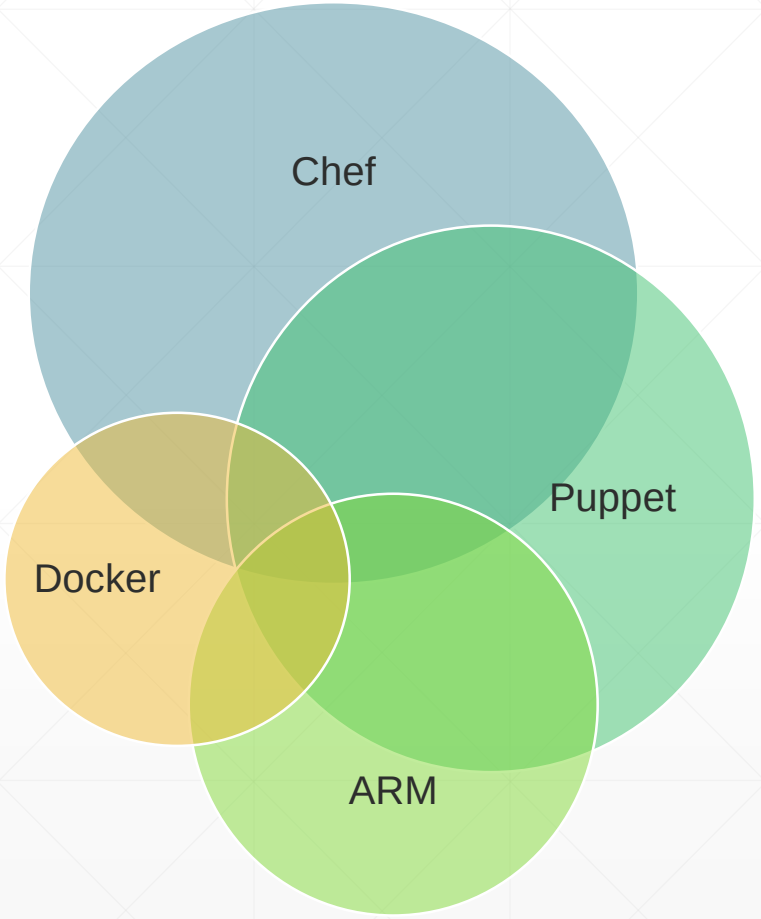
Monitoring Tools

- Stackify - https://stackify.com/
- Raygun - https://raygun.com/
- Datadog - https://www.datadoghq.com/
- Application Insights - https://azure.microsoft.com/en-us/services/application-insights
- Grafana - https://grafana.com/

Infrastructure

Configuration as code

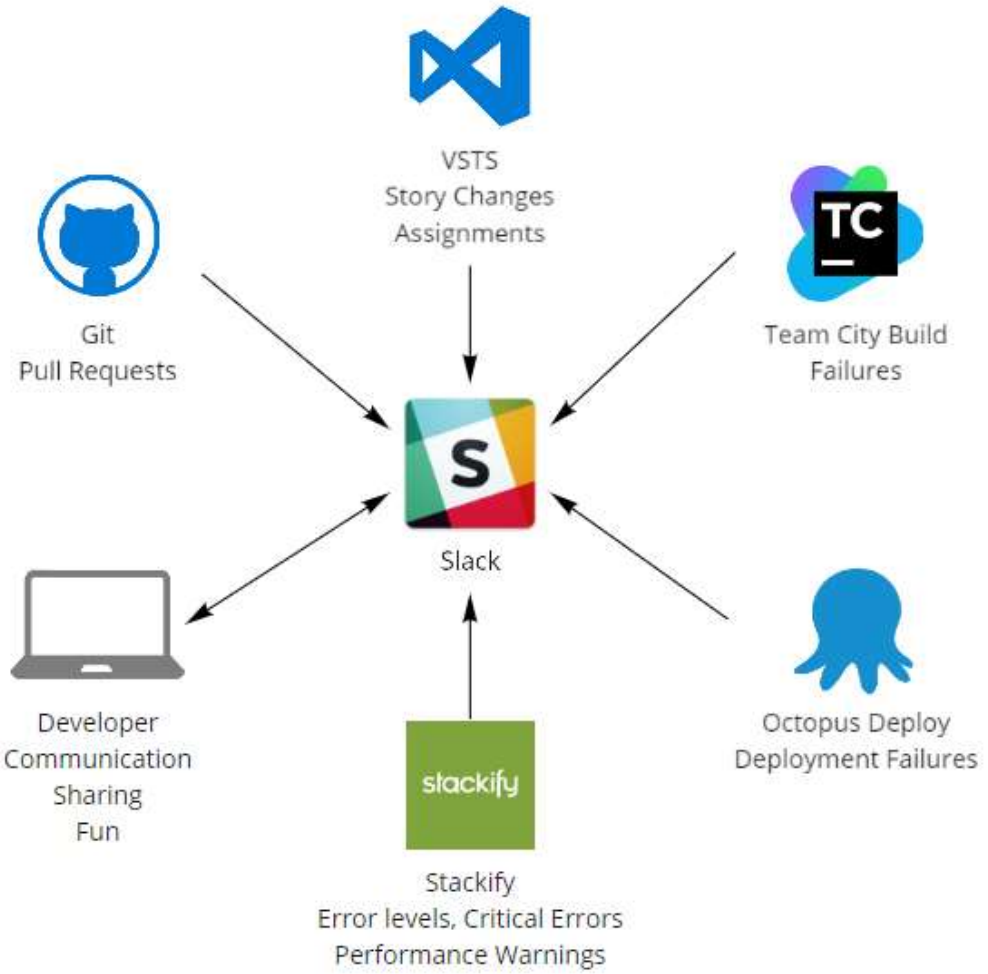
Infrastructure



- Ansible
- Salt
- Desired State Config
- Azure Automation
- Octopus Deploy

Communication

Slack



- #pullrequests
- #stories
- #builds
- #productionissues
- #team
- ChatOps?

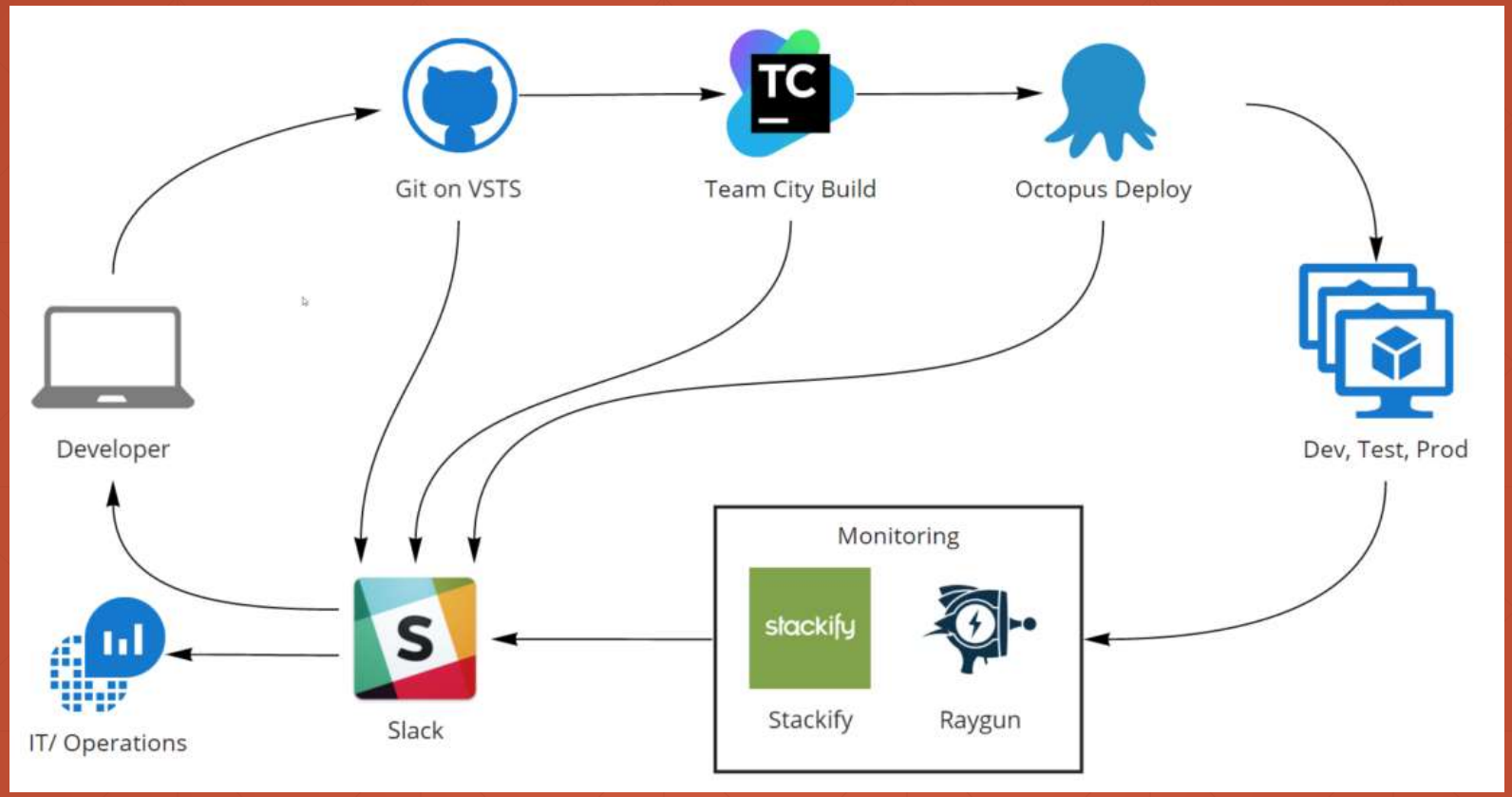
Communication with Slack

Benefits

- Transparency
- Ability to collaborate quickly and bring others in, allowing them to see the history
- One place for all notifications
- Great for releases

Learnings

- Decide on acceptable response time
- Use @mentions and @here
- Resist the urge to log everything, becomes noise
- Different channel, different urgency. Builds are right away, others can take time
- No need to keep up with everything, don’t even try, don’t expect others to
- Not the place to document decisions


Git on VSTS
Team City Build
Octopus Deploy
Developer
Dev, Test, Prod
Monitoring
stackify
Stackify
Raygun
Slack
IT/ Operations

Questions?