A simple truth in DevOps and Life

One Size Does NOT Fit All

When and where to DevOp

You probably aren't Google, Facebook or Microsoft

- Not all services need monitoring or even metrics
- Kubernetes can be both the solution to a problem and the source of many others
- Automating deployments is not always something you need
- You can design for Chaos, but make sure it matters first

But Do Make Your Designs DevOps Ready

- Make sure you use `context.Context` in every function/method
  - This allows adding things like OpenTelemetry tracing a breeze
    - Just because a function cannot be cancelled doesn't mean it shouldn't get a `Context` object
      - **I forget this sometimes and almost always regret it**
  - Gain access to authorization information at a later date
  - Pass logging objects, like the new `slog` package loggers
- Add basic metric exporting for all services
  - This requires a few lines of code
  - Metrics can be handy even if you never setup collection
- Build proof into every service
  - For a small cost you get a lot of benefit
- Prefer `Client` packages for services over straight REST or gRPC calls
  - You can upgrade all users of your service with a single change
    - Such as adding rate limiters, exponential backoff, ….
- Provide common packages that everyone builds on:
  - Base HTTP servers
  - Base RPC clients
  - Authentication/Authorization/Auditing (AAA) needs

When should you consider monitoring?

- When you have external users
- When a system failure causes failures in other systems
- When your system has complex interactions
- When your system has mission critical data that is time sensitive to users

When should you consider metrics?

- When you have external users putting real load on a system
- When you need a load-balancer to serve multiple instances for performance
- When performance issue affect critical systems
- When memory use is a factor

When should you consider Tracing?

- When you have a lot of logging outside of `main()`
- When you have a lot of users and want to log only a percentage of requests, a particular user request or set of requests

When should you consider Automation?

- When manual execution becomes a time sink
- If the task is complex, but the steps are repeatable
- When a task is dangerous to manual missteps, such as a bad copy/paste
- When the cost is low

When should you consider Orchestration

- When you are operating lots of disparate jobs
- When you have a catalog of software products
- You have hundreds of machines
- You have a machine resource underutilization and over utilization problem

Agenda

- Writing command line applications
  - Basic flag usage
  - Using argv, stdin and stdout
  - Using Cobra
  - Handling signals
- Fleet automation the hard way
  - Automating local changes with os.Exec
  - Automating remote changes with SSH
  - Creating a basic System Agent
- OTel / Observability (DJ)
  - Tracing with Events
  - Measuring with Metrics
  - Visualization and Alerting
  - Stochastic / focused sampling
- Kubernetes (DJ)
  - Introduction to Kubernetes
  - Automating deployment using the Kubernetes API
  - Customizing the Kubernetes API
- Bonus:
  - GitHub Actions (DJ)
    - Introduction to Actions
    - Authoring Actions with Go
    - Optimizing Go Actions
  - Terraform (DJ)
    - Introduction to Terraform
    - Authoring Terraform Plugins
  - Design for Chaos

Housekeeping

- Git clone the following repository:
  - https://github.com/johnsiilver/gofordevopsclass.git
- go mod download