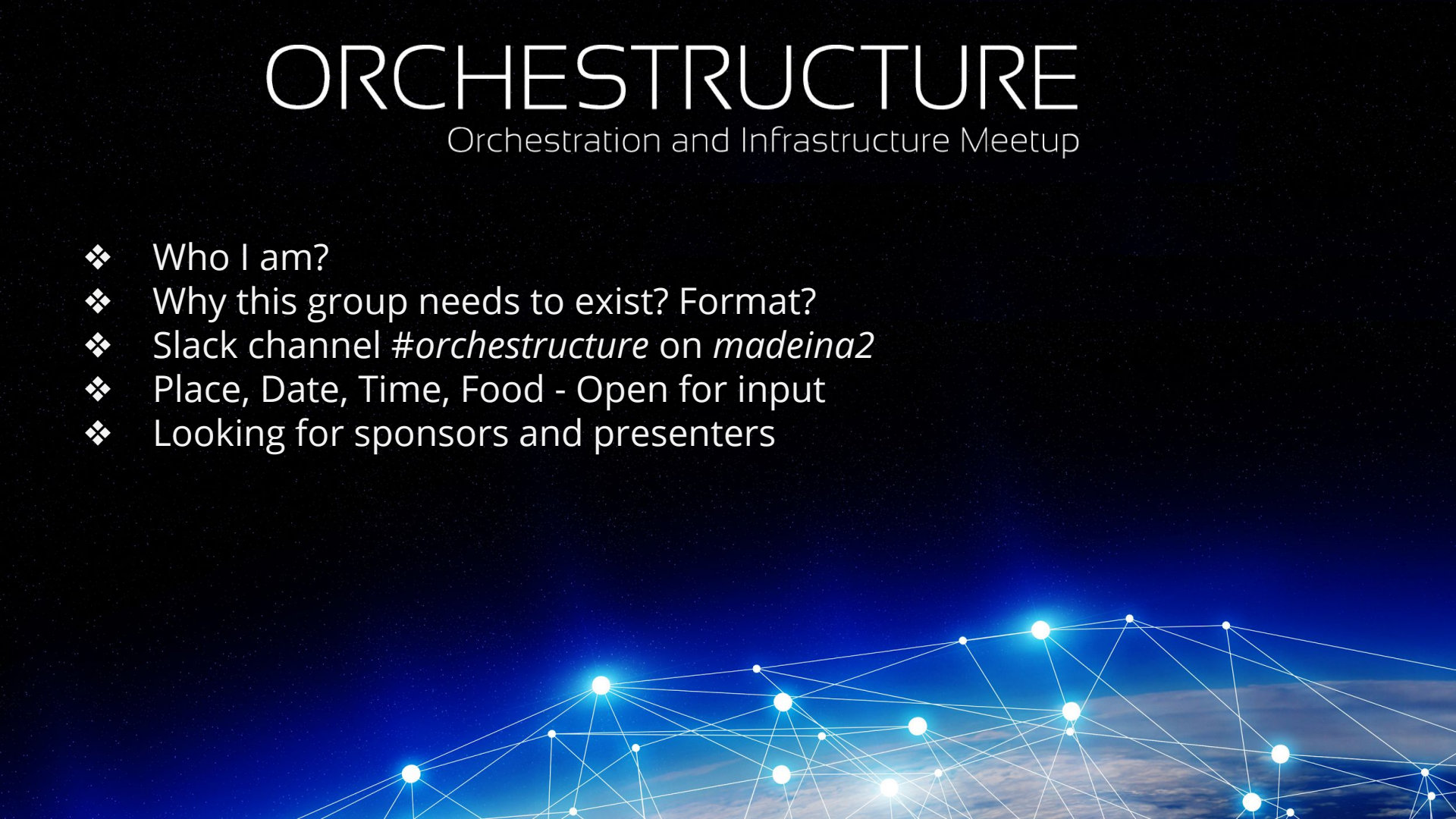
ORCHESTRUCTURE

Orchestration and Infrastructure Meetup

- Who I am?
- Why this group needs to exist? Format?
- Slack channel *#orchestructure* on *madeina2*
- Place, Date, Time, Food - Open for input
- Looking for sponsors and presenters



$ cat future.yaml

Web: "Working on basic Website for https://orchestructure.io/"

Sponsors: "Sponsors donating Infra for various uses"

Ideas: "Things like IRC Bouncer, netboot images, http testing, pastie, etc."

Social: "Made a twitter @orchestructure - Follow!"

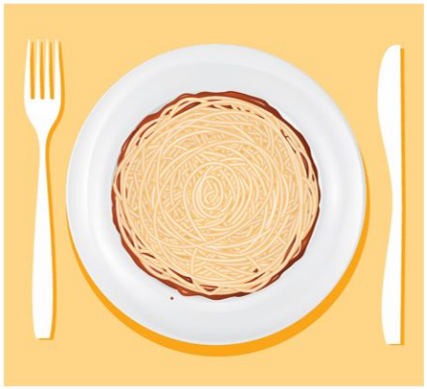
Back in Nam'

- Monolithic Applications
- Why is everything stateful?
- The "Bugzilla Server"
- "I need direct database access PLZ"
- Single Point of Failure
- Resource Wastefulness
- Meh no API br0
- Persistence/Static Env
- Mutable (lets all login!)
- It's about the server, not the service.

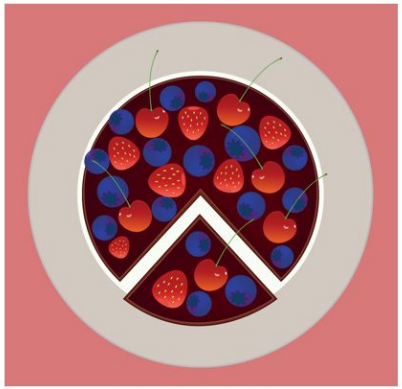
Service-Oriented Architecture

- Business Level vs Architecture Level
- Providers vs Consumers
- API's - A common interface
- Discrete Functionality
- Automated Deployment
- SaaS, PaaS, IaaS
- **Deployment methods still about the same.**

Service-Oriented Architecture



With monolithic, tightly coupled applications, all changes must be pushed at once, making continuous deployment impossible.



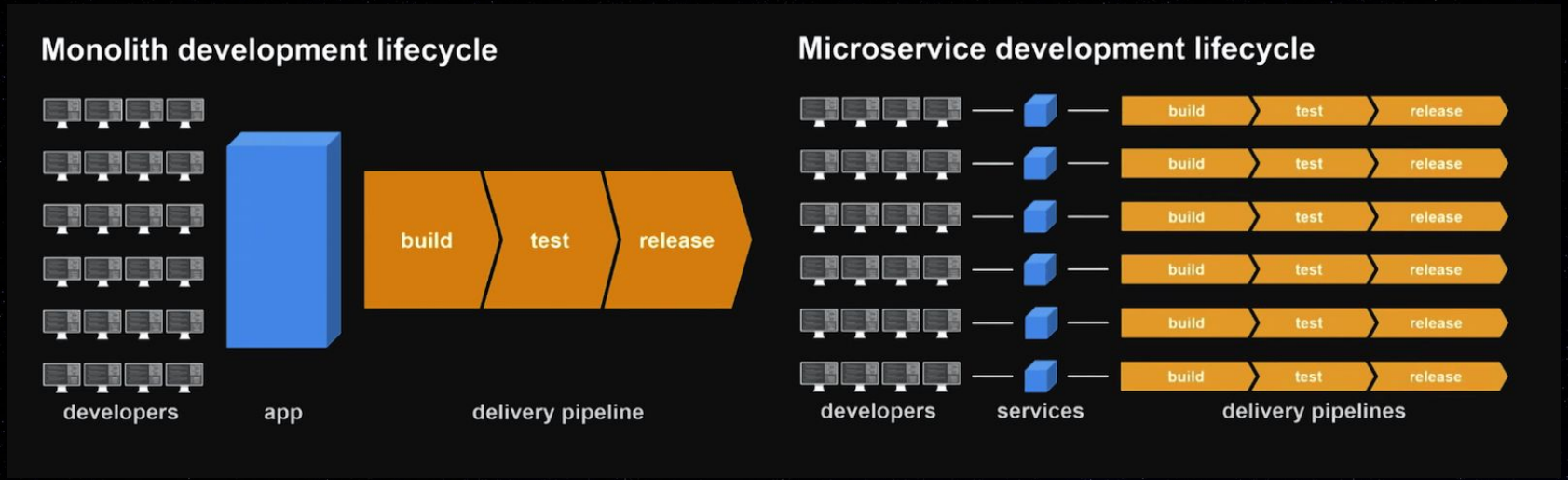
Traditional SOA allows you to make changes to individual pieces. But each piece must be carefully altered to fit into the overall design.

With a microservices architecture, developers create, maintain and improve new services independently, linking info through a shared data API.

Progression

- Packer, baking vs frying
- Configuration Management Tools
- Scripts, Manual Intervention
- 12-Factor App
- Individualism of applications (separation and isolation)
- Immutable Components
- Removing SPOF and Load Balancing
- Distributed Approach
- *App on a Server -> Service in the Cluster*

Progression



Welcome to 2017

- Microservices all the things!
- Legacy Applications though?
- Loosely Coupled Packaging (Art of the Dockerfile)
- Autoscaling, Discovery, CI/CD, Stateless Operation
- Environment Separation, Centralized API, Client Tools
- Why do I need SSH?
- Orchestration/Scheduling Engines, K/V Stores

Players

- OpenShift
- AWS ECS
- GCE
- Joyent Triton
- Bluemix

- Hyper.sh
- Kontena
- Bitnami
- Flynn
- Tsuro

- Kubernetes
- Mesos+Marathon (DC/OS)
- Rancher
- Docker Swarm

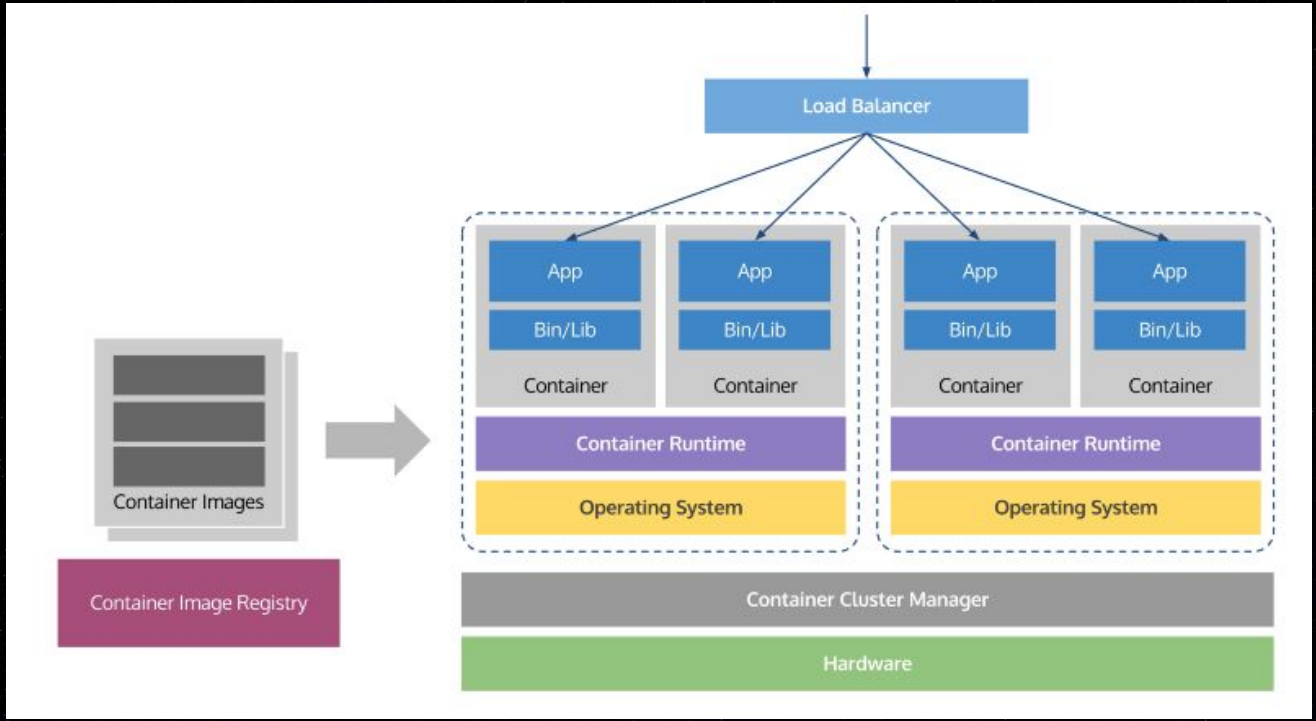
The Service

API and Tools

Foundation Services

Agent

Node



Pieces

- Storage
- Networking
- Health Checking
- Deployment Configuration
- Container Runtime
- Auto-Actions
- Pipeline
- Load Balancing
- Scheduling
- Ephemerality and Immutability
- Messaging/Logging/Events

```
spec:
  containers:
  - name: "arroyo-sentry-web"
    image: registry.gitlab.com/arroyonetworks/arroyo-sentry
    args:
    - "run"
    - "web"
    env:
      - name: "SENTRY_DB_USER"
        value: "sentry"
      - name: "SENTRY_POSTGRES_HOST"
        value: "postgres.development.svc.cluster.local"
      - name: "SENTRY_REDIS_HOST"
        value: "127.0.0.1"
      - name: "SENTRY_REDIS_PASSWORD"
        value: "nopenopenope"
    ports:
    - name: http
      containerPort: 9000
  - name: "arroyo-sentry-worker"
    image: registry.gitlab.com/arroyonetworks/arroyo-sentry
    args:
    - "run"
    - "worker"
  - name: "arroyo-sentry-cron"
    image: registry.gitlab.com/arroyonetworks/arroyo-sentry
    args:
    - "run"
    - "cron"
  - name: "arroyo-sentry-redis"
    image: redis:3.2-alpine
    args:
    - "redis-server"
    - "--requirepass"
    - "nopenopenope"
    ports:
    - name: redis
      containerPort: 6379
  - name: "backup"
    image: inanimate/s3-archive
    env:
    - name: "AWS_ACCESS_KEY_ID"
      value: "lulz"
    - name: "AWS_SECRET_ACCESS_KEY"
      value: "sweetkeybr0"
    - name: "BUCKET"
      value: "s3://backups/sentry"
    - name: "NAME_PREFIX"
      value: "sentry-data-backup"
    - name: "SYMMETRIC_PASSPHRASE"
      value: "supercoolreliablebackup"
  - name: "fly.io-wormhole"
    image: "flyio/wormhole:0.5.24"
    env:
    - name: "FLY_TOKEN"
      value: "x"
    - name: "FLY_LOCAL_ENDPOINT"
      value: "127.0.0.1:9000"
```

Key Takeaways

- Nodes are becoming Ephemeral (so is the Cluster)
- The Commit ID (or Tag) is King End-to-End
- Configuration Contained - IaC
- Away with the Monolith
- Test Early and Often (CI/CD Pipelines)
- Stateless all the way
- Input, Processing, Output
- Utilize Upstream as much as possible
- Expect and Plan for Failure
- Distributed Self-Healing Applications

Quoted

"Microservices impose real, often underestimated costs. But I think those costs can actually be thought of as virtuous constraints. When you design a distributed system, you open yourself up to a lot of errors. But it's a lot better to defend against those errors from day 1, than to try to bolt a defense on to an existing system later. And in our zeitgeist, you're going to have to think distributed at some point."

Peter Bourgon

Other Resources

- https://github.com/veggiemonk/awesome-docker
- https://12factor.net/
- http://microservices.io/patterns/microservices.html
- https://www.fastly.com/blog/microservices-war-stories
- https://thenewstack.io/microservices-changed-matter/
- https://peter.bourgon.org/a-case-for-microservices/
- https://read.acloud.guru/evolution-of-business-logic-from-monoliths-through-microservices-to-functions-ff464b95a44d
- /r/kubernetes, /r/docker, /r/devops
- @kelseyhightower, @jessfraz, @ibuildthecloud, @timothysc, @brandonphilips, @jbeda, @jpetazzo
- Youtube Talks (dockercon, kubecon, hashiconf, googlenext)

Future Presentations

- More on Netflix, Imgur, and other architectures
  - *https://www.slideshare.net/stonse/microservices-at-netflix*
- Eggs in One Basket: Dodging single points of failure
  - *https://blog.sentry.io/2017/03/01/dodging-s3-downtime-with-nginx-and-haproxy.html*
- Security in a Containerized World: Handling Secrets
  - *A look at Vault, Docker and k8s secrets.*
- Dynamic SSL Certificates: Managed LetsEncrypt
- Rancher's Josh Curl Presenting on Rancher (June?)


There is no cloud
It's just someone else's computer