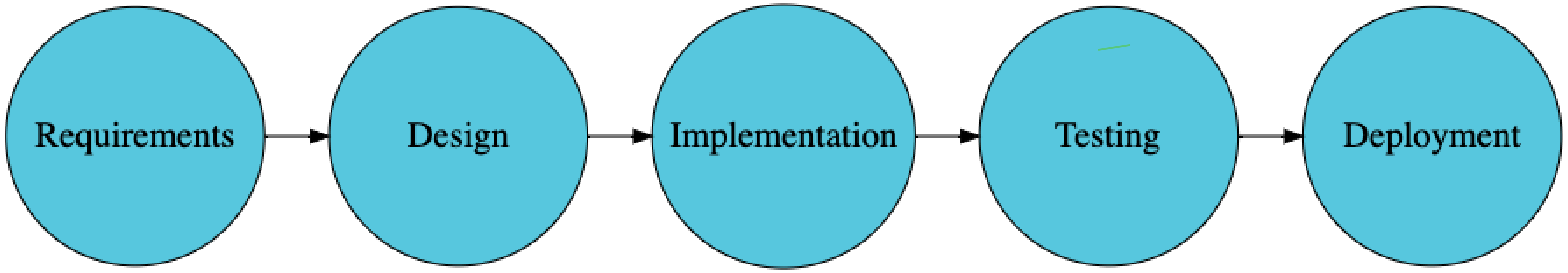
Evolution

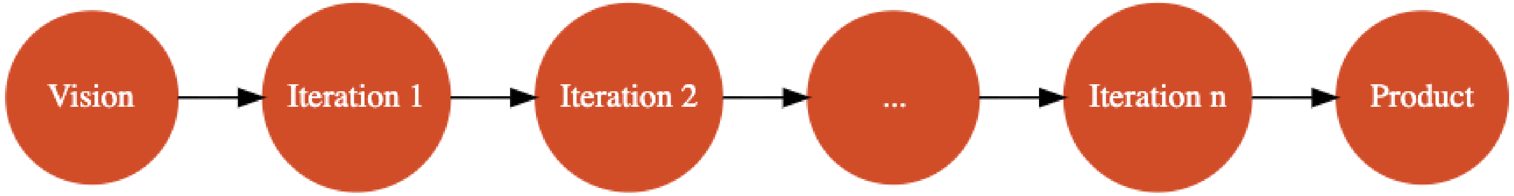
Agile > DevOps > SRE

Software development life cycle (SDLC) - Waterfall



- Software development in multiple long phases:
  - Requirements
  - Design
  - Implementation
  - Testing
  - Deployment

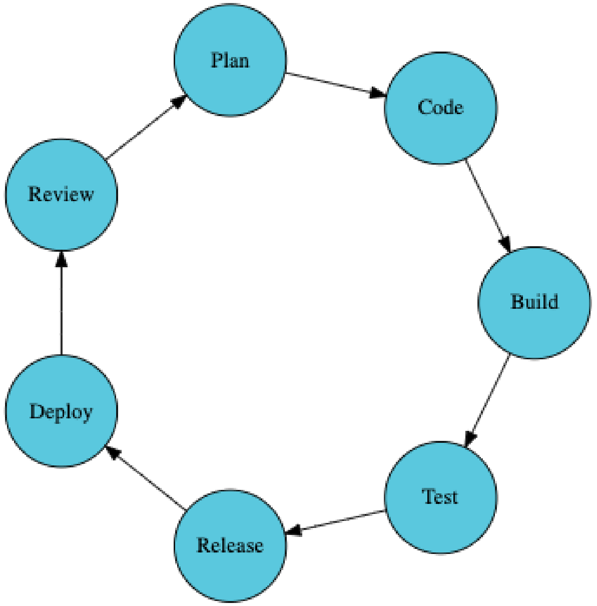
Software development life cycle (SDLC) - Spiral



- **Software development in smaller iterations:**
  - Start
  - Iteration 1
  - Iteration 2
  - ...
  - ...

Software development life cycle (SDLC) - Agile

- Principles
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
  - **Now there are 12 principles** (***https://agilemanifesto.org/principles.html***)
- **Agile is recommended for most software development**:
  - BUT add a bit of rigidity from waterfall model for critical safety software (Flight navigation software, Medical devices software etc)



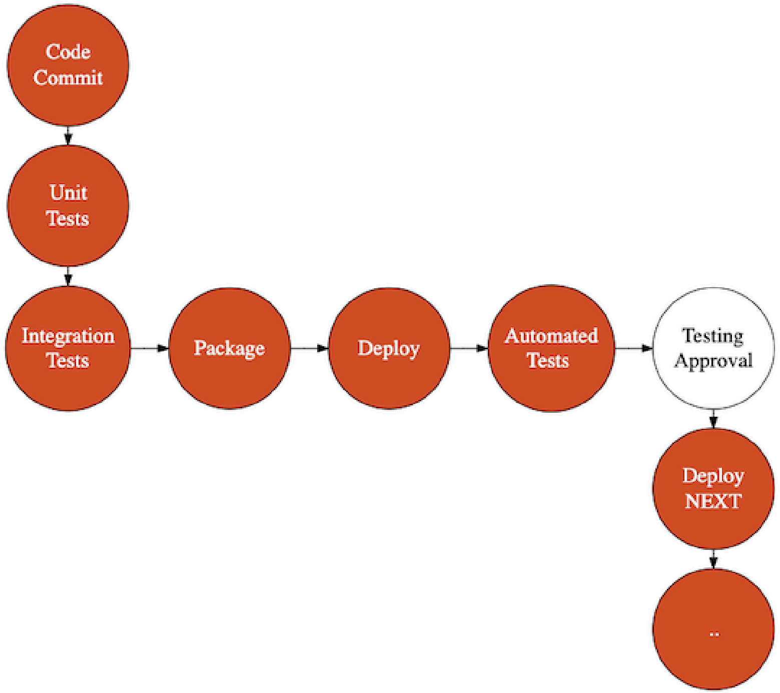
DevOps

Business
Development
Operations

- Getting Better at "**Three Elements of Great Software Teams**"
  - **Communication** - Get teams together
  - **Feedback** - Earlier you find a problem, easier it is to fix
  - **Automation** - Automate testing, infrastructure provisioning, deployment, and monitoring

DevOps - CI, CD

- **Continuous Integration**
  - Continuously run your tests and packaging
- **Continuous Deployment**
  - Continuously deploy to test environments
- **Continuous Delivery**
  - Continuously deploy to production



DevOps - CI CD - Recommended Things to Do

- **Static Code Analysis**
  - Lint, Sonar
  - Including Static Security Checks (Source Code Security Analyzer software like Veracode or Static Code Analyzer)
- **Runtime Checks**
  - Run Vulnerability Scanners (automated tools that scan web applications for security vulnerabilities)
- **Tests**
  - Unit Tests (JUnit, pytest, Jasmine etc)
  - Integration Tests (Selenium, Robot Framework, Cucumber etc)
  - System Tests (Selenium, Robot Framework, Cucumber etc)
  - Sanity and Regression Tests

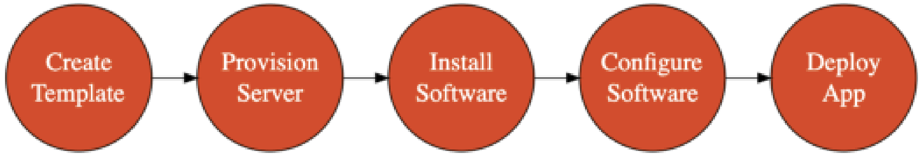
DevOps - CI, CD Tools

- **Cloud Source Repositories**: Fully-featured, private Git repository
  - Similar to Github
- **Container Registry**: Store your Docker images
- **Jenkins**: Continuous Integration
- **Cloud Build**: Build deployable artifacts (jars or docker images) from your source code and configuration
- **Spinnaker**: Multi-cloud continuous delivery platform
  - Release software changes with high velocity and confidence
  - Supports deployments to Google Compute Engine, Google Kubernetes Engine, Google App Engine and other cloud platforms
  - Supports Multiple Deployment Strategies

Container Registry

Cloud Source Repositories

DevOps - Infrastructure as Code



- Treat **infrastructure the same way as application code**
- Track your infrastructure **changes over time** (version control)
- Bring **repeatability** into your infrastructure
- Two Key Parts
  - **Infrastructure Provisioning**
    - Provisioning compute, database, storage and networking
    - Open source cloud neutral - Terraform
    - GCP Service - Google Cloud Deployment Manager
  - **Configuration Management**
    - Install right software and tools on the provisioned resources
    - Open Source Tools - Chef, Puppet, Ansible and SaltStack

Google Cloud Deployment Manager - Introduction

- Lets consider an example:
  - I would want to create a new VPC and a subnet
  - I want to provision a Load balancer, Instance groups with 5 Compute Engine instances and an Cloud SQL database in the subnet
  - I would want to setup the right Firewall
- AND I would want to create 4 environments
  - Dev, QA, Stage and Production!
- Deployment Manager can help you do all these with a simple (actually NOT so simple) script!

Deployment Manager

Google Cloud Deployment Manager - Advantages

- Automate deployment and modification of Google Cloud resources in a controlled, predictable way
  - Deploy in multiple environments easily!
- Avoid configuration drift
- Avoid mistakes with manual configuration
- Think of it as version control for your environments
- **Important Note** - Always modify the resources created by Deployment Manager using Deployment Manager

Deployment Manager

Google Cloud Deployment Manager

In28 Minutes

- All configuration is defined in a simple text file - YAML
  - I want a VPC, a subnet, a database and ...
- Deployment Manager understands dependencies
  - Creates VPCs first, then subnets and then the database
- (Default) Automatic rollbacks on errors (Easier to retry)
  - If creation of database fails, it would automatic delete the subnet and VPC
- Version control your configuration file and make changes to it over time
- Free to use - Pay only for the resources provisioned
  - Get an automated estimate for your configuration

Deployment Manager

Cloud Deployment Manager - Example

```
- type: compute.v1.instance
    name: my-first-vm
    properties:
        zone: us-central1-a
        machineType: <<MACHINE_TYPE>>
        disks:
        - deviceName: boot
            type: PERSISTENT
            boot: true
            autoDelete: true
            initializeParams:
                sourceImage: <<SOURCE_IMAGE>>
        networkInterfaces:
        - network: <<NETWORK>>
            # Give instance a public IP Address
            accessConfigs:
            - name: External NAT
                type: ONE_TO_ONE_NAT
```

Cloud Deployment Manager - Terminology

- **Configuration** file: YAML file with resource definitions for a single deployment
- **Templates**: **Reusable resource definitions** that can be used in multiple configuration files
  - Can be defined using:
    - Python (preferred) OR
    - JinJa2 (recommended only for very simple scripts)
- **Deployment**: Collection of resources that are deployed and managed together
- **Manifests**: Read-only object containing original deployment configuration (including imported templates)
  - Generated by Deployment Manager
  - Includes fully-expanded resource list
  - Helpful for troubleshooting

Cloud Marketplace (Cloud Launcher)

- Installing custom software might involve setting up multiple resources:
  - Example: Installing WordPress needs set up of compute engine and a relational database
- How do you simplify the set up of custom software solutions like Wordpress or even more complex things like SAP HANA suite on GCP?
- **Cloud Marketplace**: Central repo of easily deployable apps & datasets
  - Similar to **App Store/Play Store** for mobile applications
  - You can search and install a complete stack
    - Commercial solutions - SAP HANA etc
    - Open Source Packages - LAMP, WordPress, Cassandra, Jenkins etc
    - OS Licenses: BYOL, Free, Paid
    - Categories: Datasets/Developer tools/OS etc
  - When selecting a solution, you can see:
    - Components - Software, infrastructure needed etc
    - Approximate price

Site Reliability Engineering (SRE)

Google Cloud

- **DevOps++ at Google**
- SRE teams **focus on every aspect of an application**
  - availability, latency, performance, efficiency, change management, monitoring, emergency response, and capacity planning
- **Key Principles:**
  - Manage by Service Level Objectives (SLOs)
  - Minimize Toil
  - Move Fast by Reducing Cost of Failure
  - Share Ownership with Developers

Site Reliability Engineering (SRE) - Key Metrics

- **Service Level Indicator(SLI):** Quantitative measure of an aspect of a service
  - Categories: availability, latency, throughput, durability, correctness (error rate)
  - Typically aggregated - "Over 1 minute"
- **Service Level Objective (SLO)** - SLI + target
  - 99.99% Availability, 99.999999999% Durability
  - Response time: 99th percentile - 1 second
  - Choosing an appropriate SLO is complex
- **Service Level Agreement (SLA):** SLO + consequences (contract)
  - What is the consequence of NOT meeting an SLO? (Defined in a contract)
  - Have stricter internal SLOs than external SLAs
- **Error budgets:** (100% – SLO)
  - How well is a team meeting their reliability objectives?
  - Used to manage development velocity

Site Reliability Engineering (SRE) - Best Practices

- Handling Excess Loads
  - Load Shedding
    - API Limits
      - Different SLAs for different customers
    - Streaming Data
      - If you are aggregating time series stream data, in some scenarios, you can drop a part of data
  - Reduced Quality of Service
    - Instead of talking to a recommendations API, return a hardcoded set of products!
    - Not always possible:
      - Example: if you are making a payment
- Avoiding Cascading Failures
  - Plan to avoid thrashing
    - Circuit Breaker
    - Reduced Quality of Service

Google Cloud

Site Reliability Engineering (SRE) - Best Practices - 2

Google Cloud

- **Penetration Testing (Ethical Hacking)**
  - Simulate an attack with the objective of finding security vulnerabilities
  - Should be authorized by project owners
  - No need to inform Google
    - Ensure you are only testing your projects and are in compliance with terms of service!
  - Can be white box (Hacker is provided with information about infrastructure and/or applications) or black box (No information is provided)
- **Load Testing** (JMeter, LoadRunner, Locust, Gatling etc)
  - Simulate real world traffic as closely as possible
  - Test for spiky traffic - suddenly increases in traffic

Site Reliability Engineering (SRE) - Best Practices - 3

- **Resilience Testing** - "How does an application behaves under stress?"
- **Resilience** - "Ability of system to provide acceptable behavior even when one or more parts of the system fail"
- **Approaches:**
  - **Chaos Testing (Simian Army)** - cause one or more layers to fail
    - "unleashing a wild monkey with a weapon in your data center to randomly shoot down instances and chew through cables"
  - Add huge stress on one of the layers
  - **Include network in your testing** (VPN, Cloud Interconnect etc..)
    - Do we fall back to VPN if direct interconnect fails?
    - What happens when internet is down?
  - **Best Practice: DiRT** - disaster recovery testing at Google
    - Plan and execute outages for a defined period of time
    - Example: Disconnecting complete data center

Google Cloud