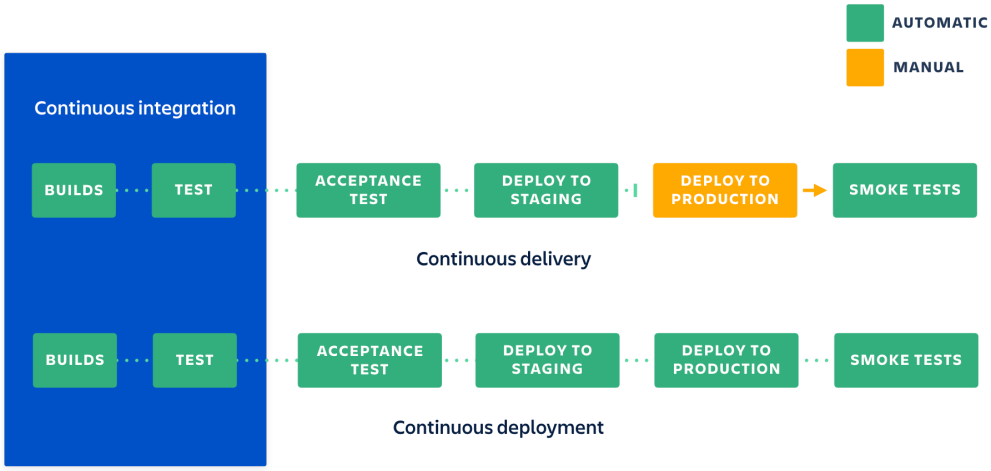
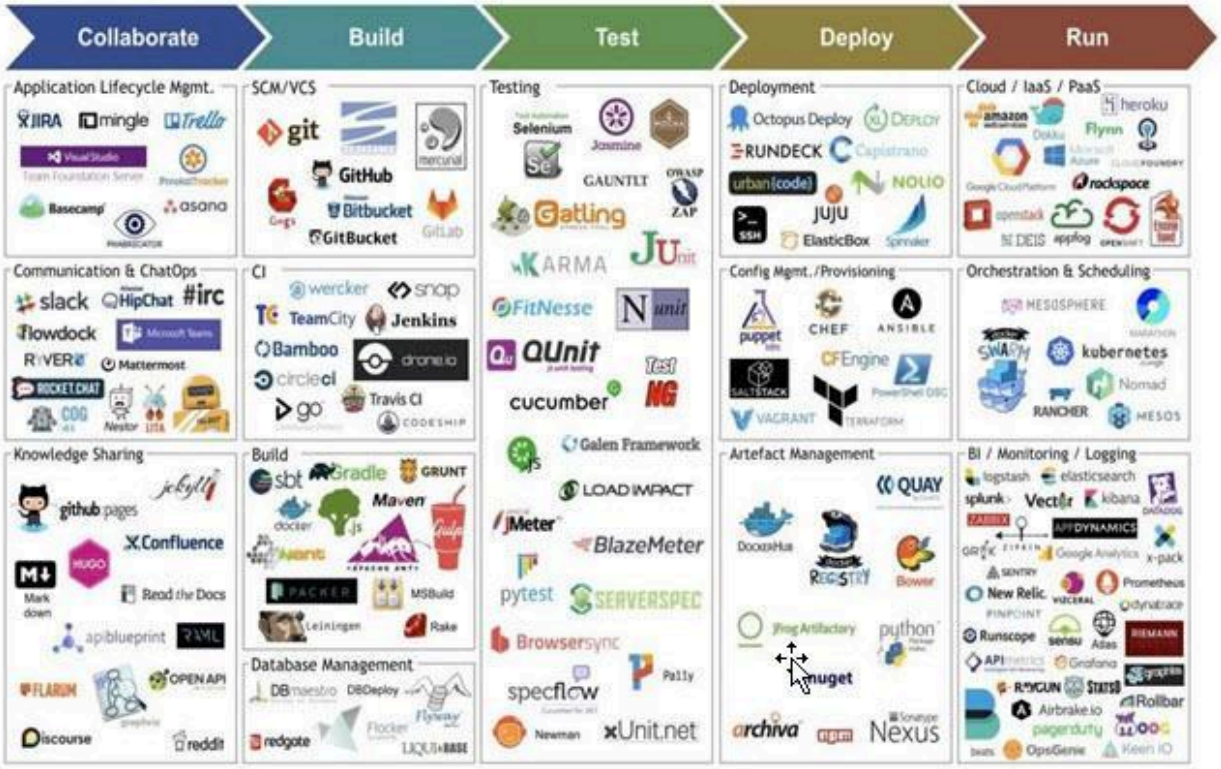
**Continuous Integration:**
Automated testing of built code

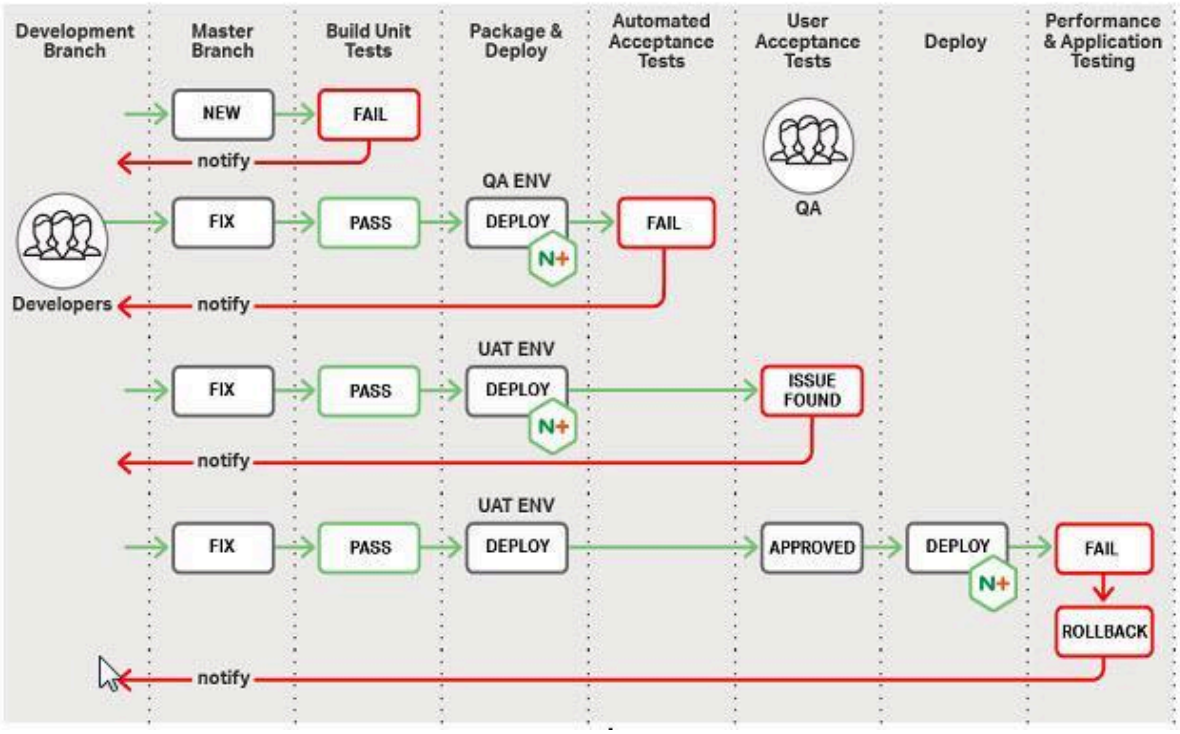
**Continuous Delivery:**
Automated release/deployment (but manual or non-automatic deploy to prod)

**Continuous Deployment**:
Automated release to production



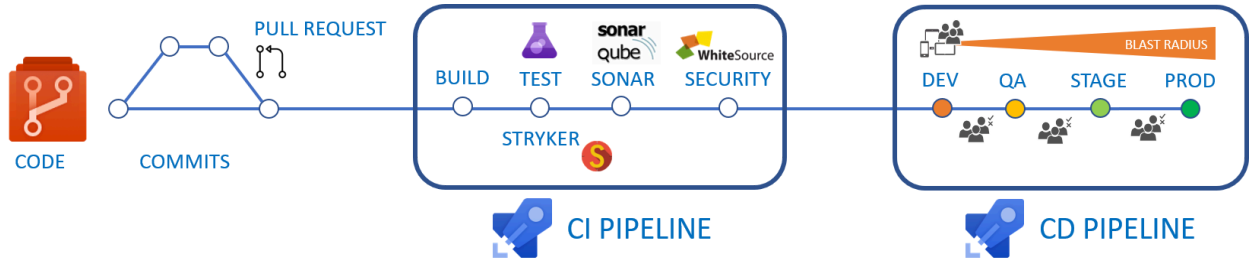

*https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment*



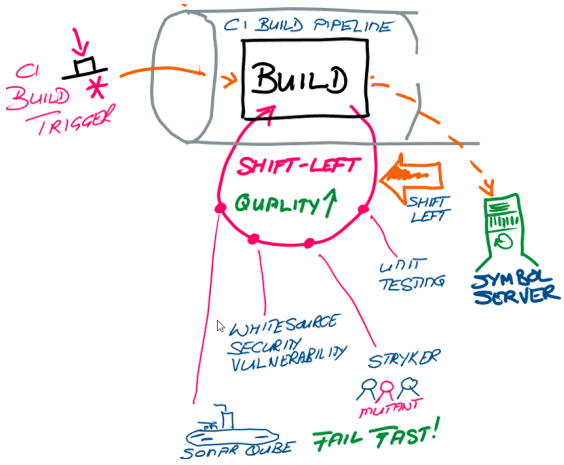




https://www.devopsuniversity.org/cicd-and-devops/

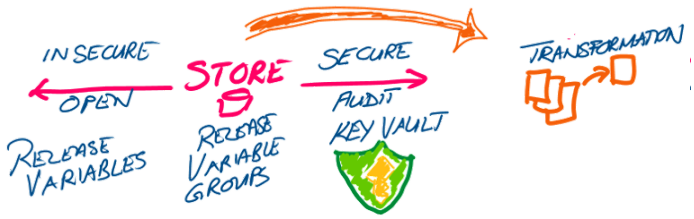
https://opensource.com/article/19/7/cicd-pipeline-rule-them-all - Explains “one commit == one artifact” well.



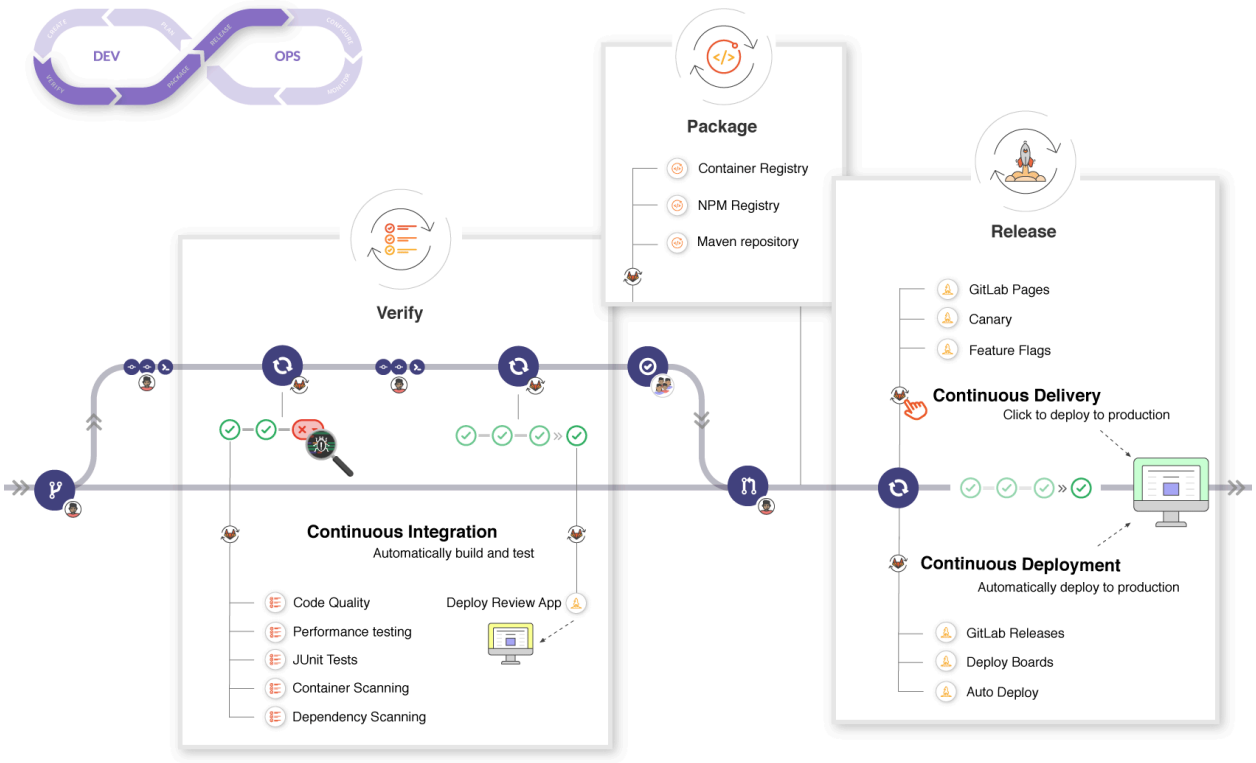

**Shift artifact validation left:**

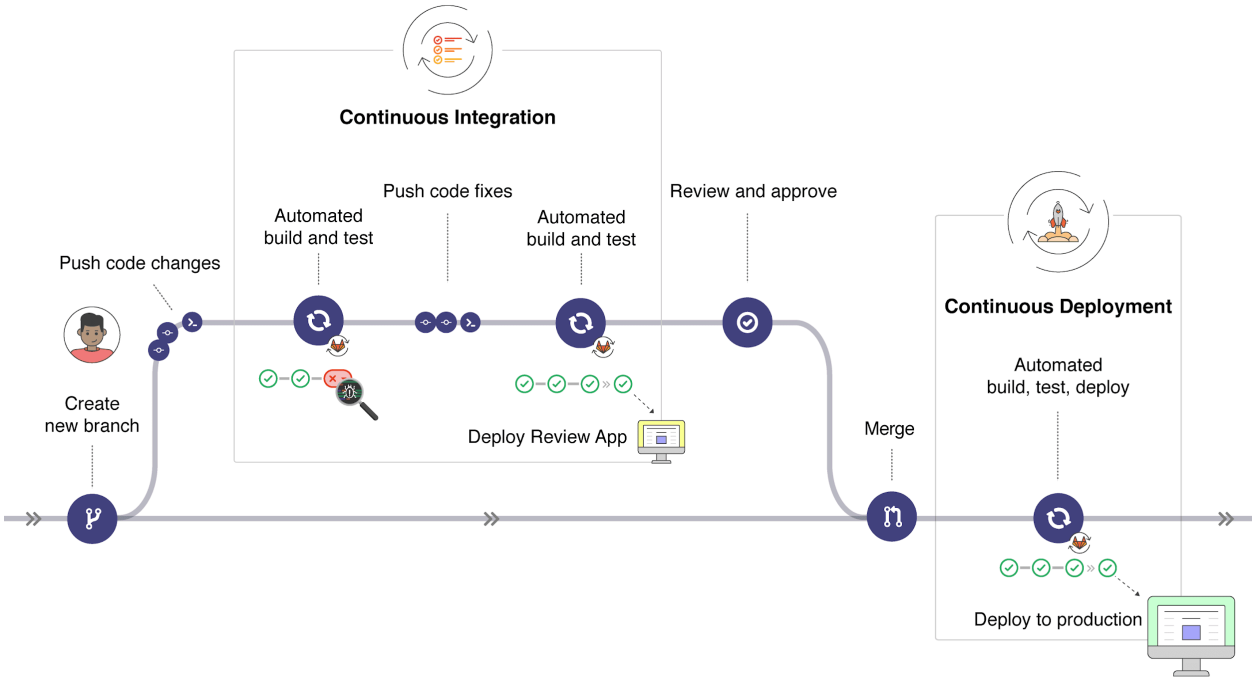



**Shift env config right:**



## Development Workflow



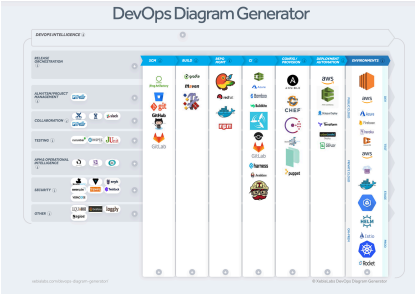




https://docs.gitlab.com/ee/ci/introduction/

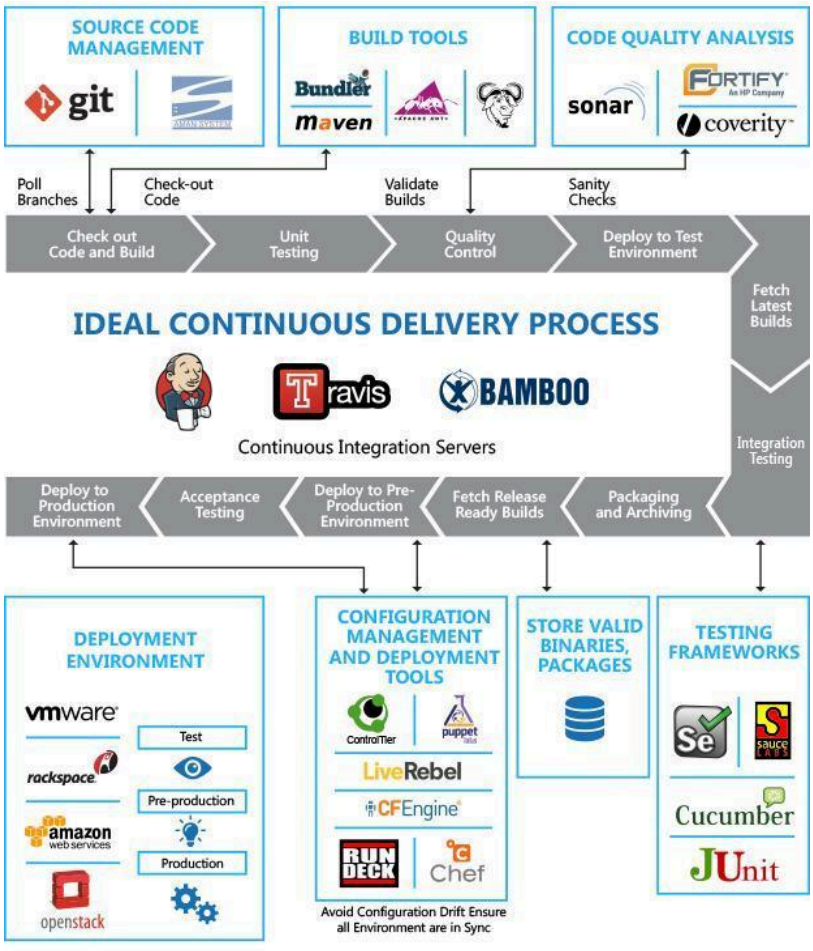
DevOps Tools Explained:

- https://www.padok.fr/en/blog/devops-tools-most-used
- https://digital.ai/periodic-table-of-devops-tools

xebialabs custom devops diagram (you can put in fake info to get past the first screen)




https://www.tmasolutions.com/devops

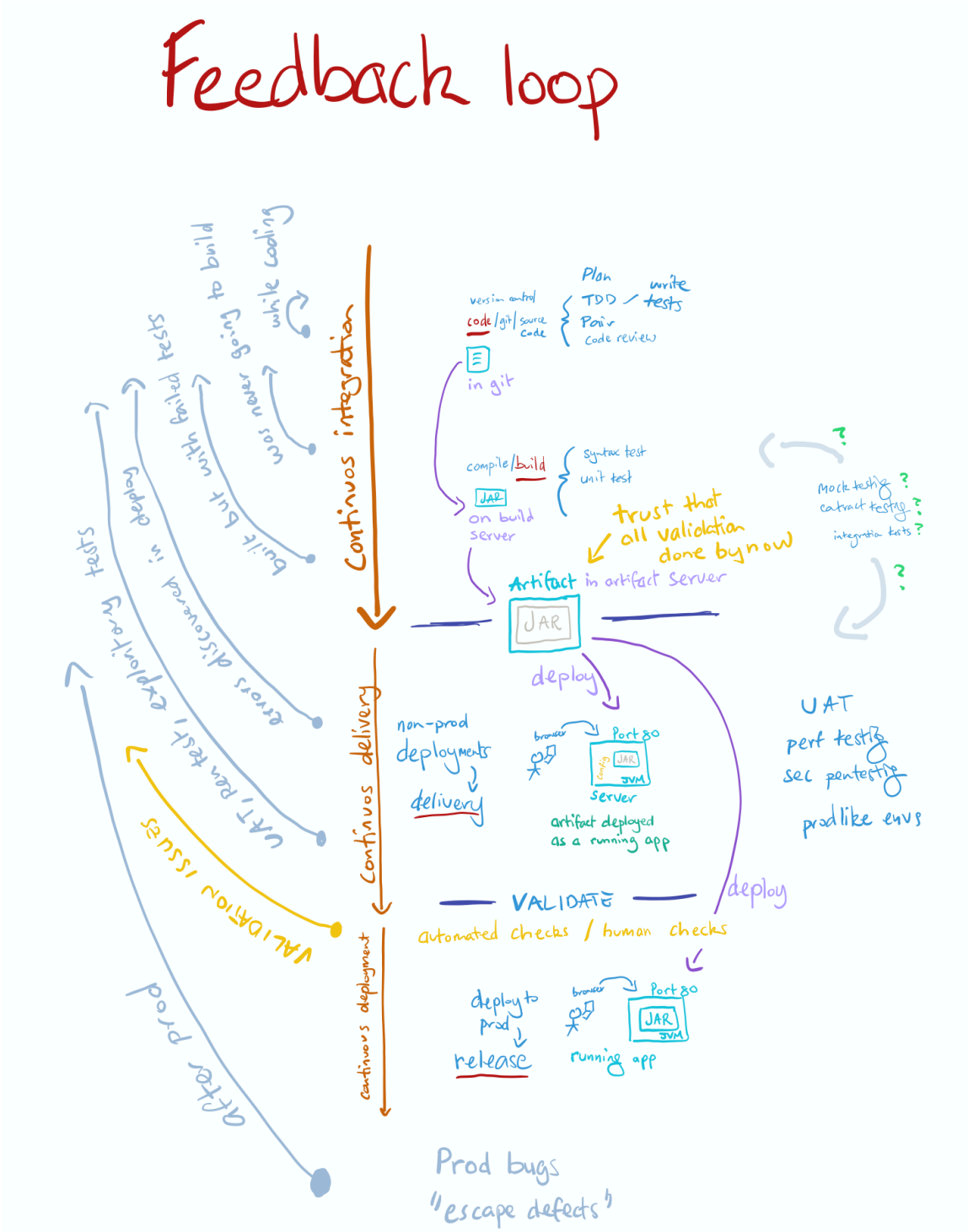


More Info:

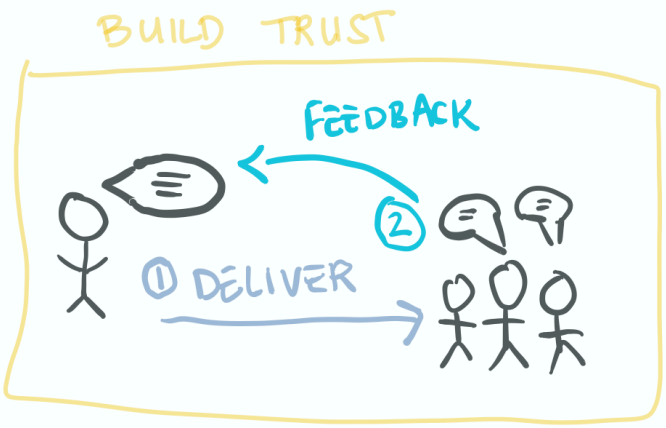
https://continuousdelivery.com/2014/02/visualizations-of-continuous-delivery/
https://app.lucidchart.com/documents/edit/c3dfdebc-a293-4900-86ef-06138f73f119/0_0

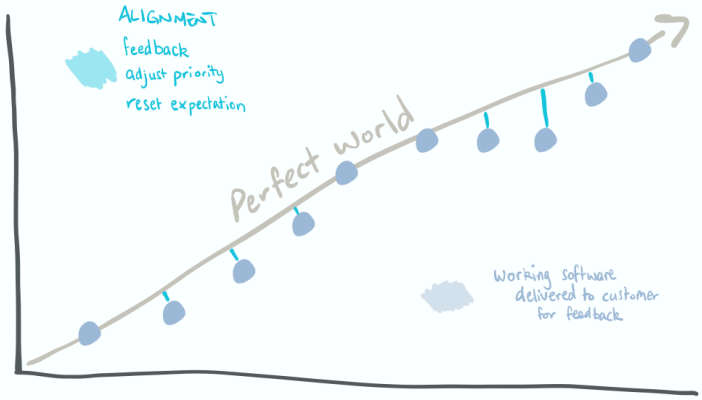
…. Everything below is extras ...

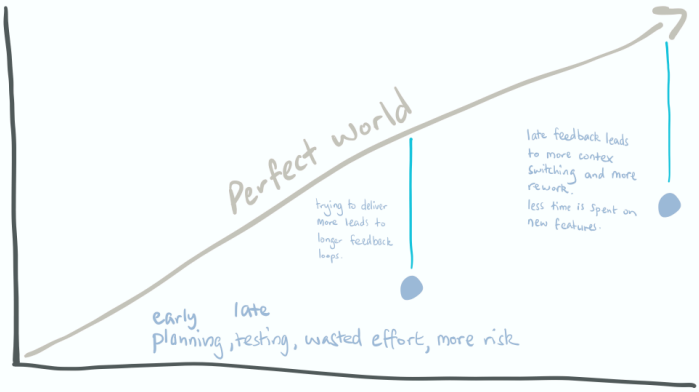
**Why do we care about fast feedback?**



why feedback matters

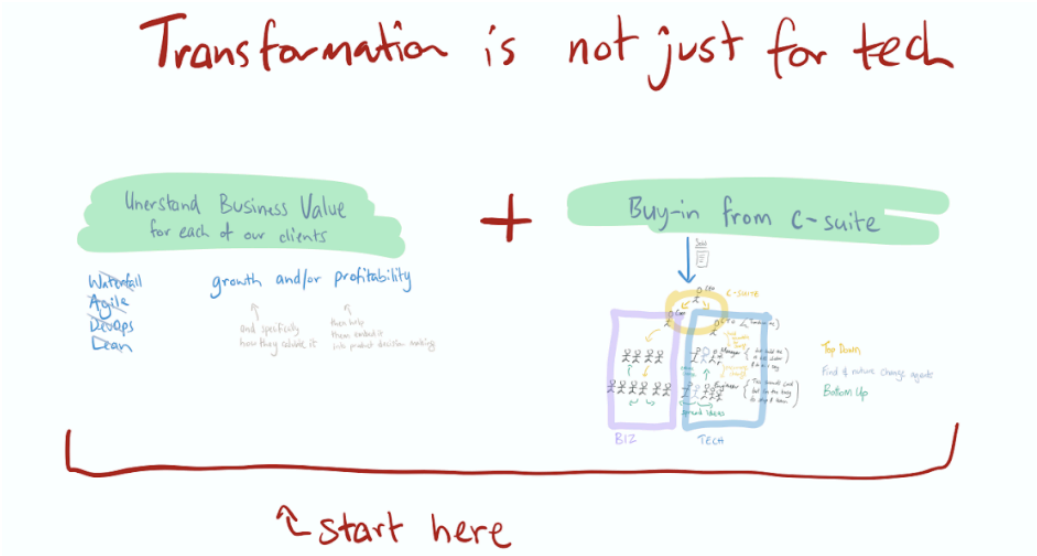


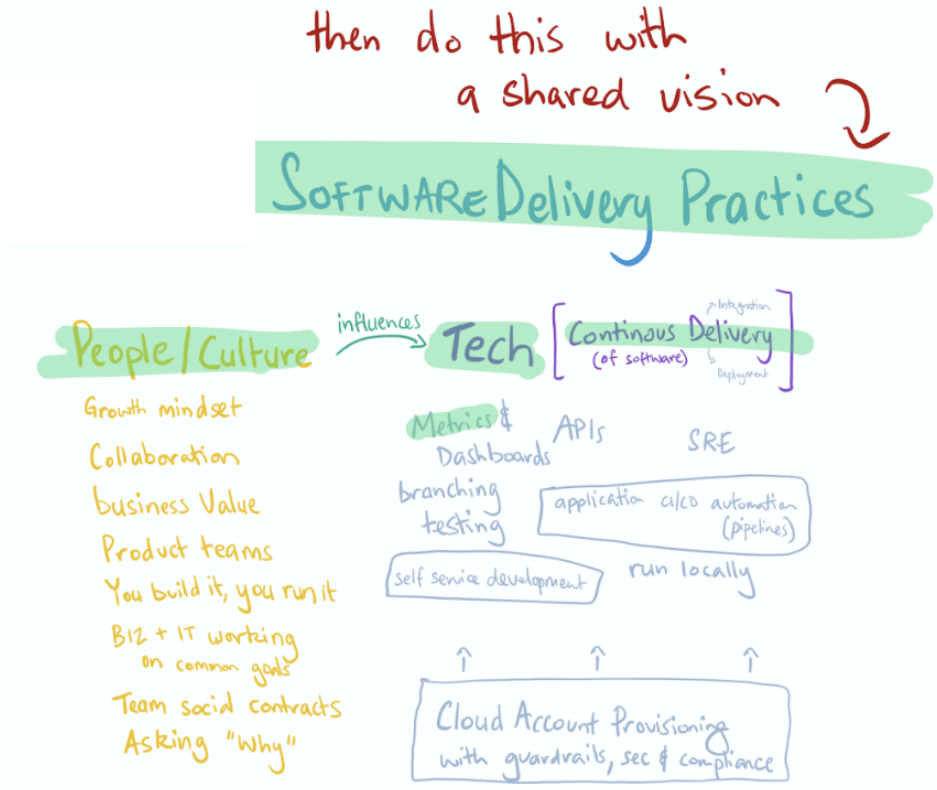



**Transformation & Development**

4 ways to explain transformation:



# Transformation Outline

Potential topics

## People

Have a Growth mindset (instead of fixed)

Continuously improve

Encourage innovation

Foster collaboration

- Growth Mindset
- Encourage innovation & collaboration
- Continuous Improvement
- Empowered teams
- No key person deps.
- Product teams ("trust hardware" "you build it, you run it") → Understand business value → short feedback loop w/ customer
- Agile/Scrum/Kanban/XP → Don't be overly prescriptive. Understand why you're doing what you're doing
- Demo Days, Office Hours, Slack, Social Contracts, High Performing teams (trust)

## Measure Everything

Make decisions based on data

Capture metrics
- for team
- for software delivery

Make metrics available
- via API
- on a dashboard
- on an actual TV

Any software or service you produce must have an API, & status/health page

Maturity Assessments
- CI/CD
- 12F
- Data
- 6R's treatment plan
- Product Delivery "Agile"

SRE

## Continuous Delivery

Higher quality software through...
repeatable, predictable delivery

Reduction of defects

Reduce cycle time:
ideation (I have an idea)
to production (released in prod)

- Trunk Based Dev
- Ability to run pipeline pre-commit
- Mocked data to test before prod
- Isolated test env (create as needed vs single "shared" envs)

Reduce friction in Cloud...
- Fully automate CloudAccount Provisioning
- Build guardrails into Cloud Account ← your account operations are your FullSafe
- Add checks in pipeline ← your pipeline checks (cattle security rules in dev)
- Let devs write code & create pipelines in as many cloud accounts as they need

↑ now you have safe self-service

- Everything driven via a pipeline, ReadOnly by default (breakglass for write access or ssh)

- containers are easy to migrate
- lambdas are easy to migrate
- Serverless is defered work, not vendor lock-in

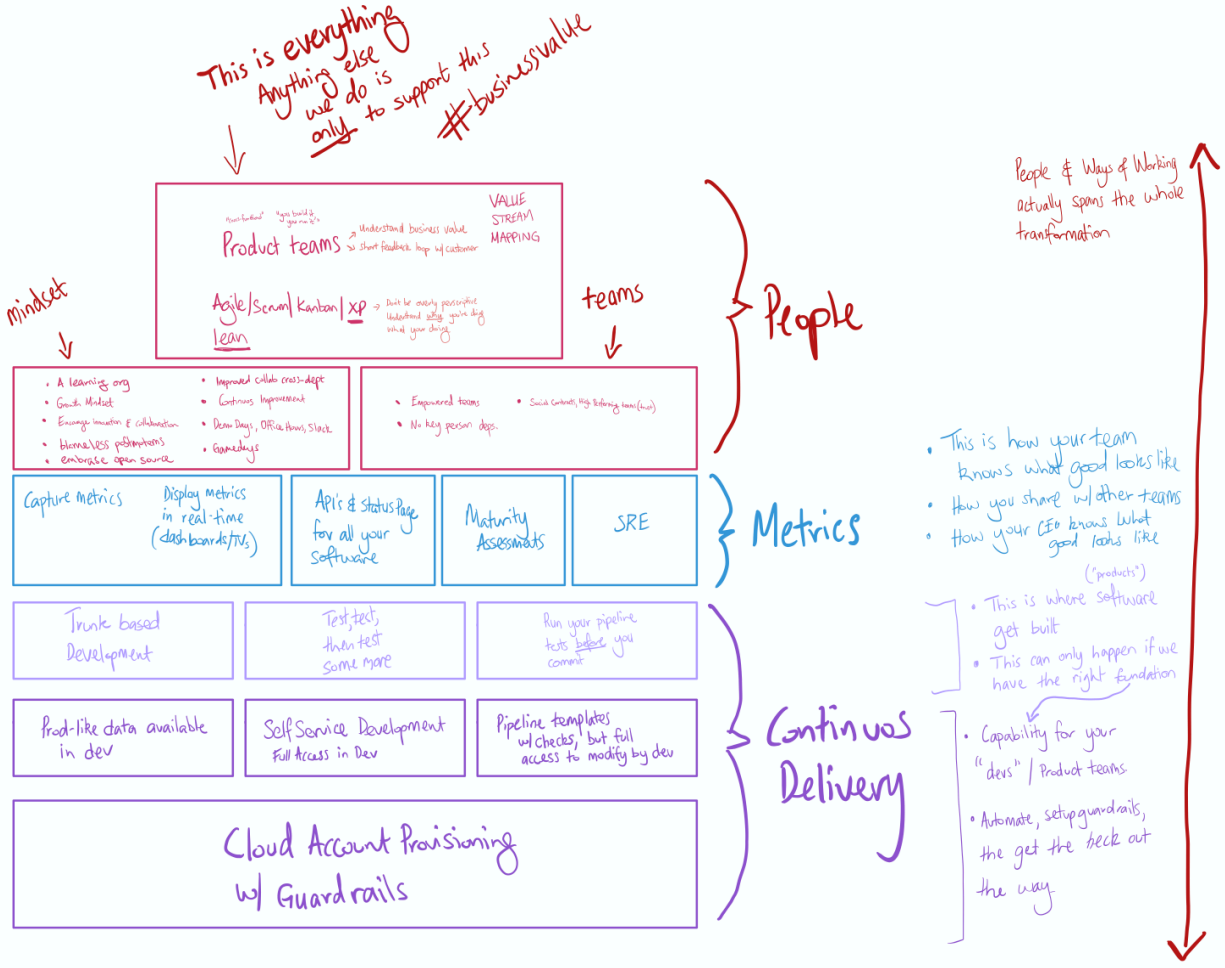
→

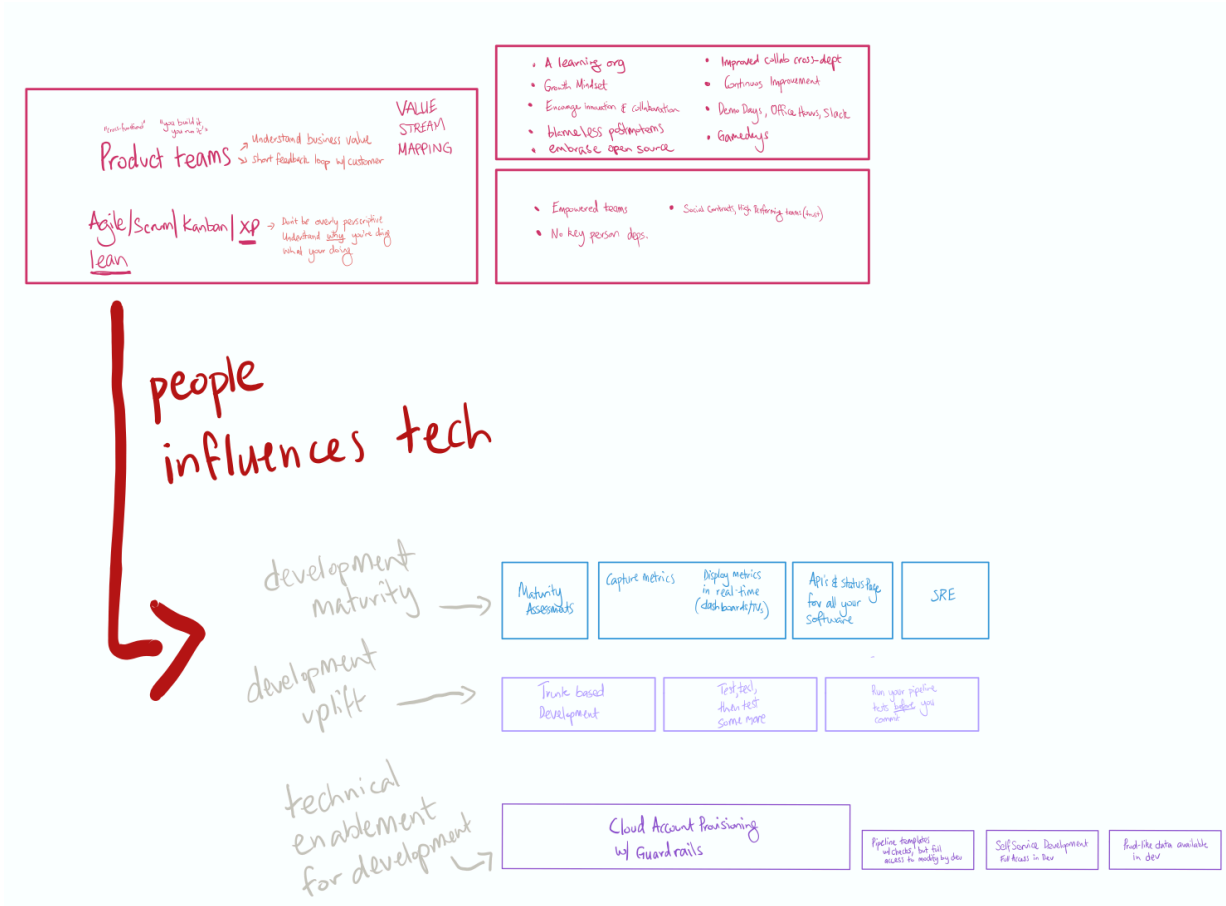
- Don't go down cloud agnostic rabbit hole
- Don't try to dictate or plan your entire architecture

↑ guidance is good, start with 12F

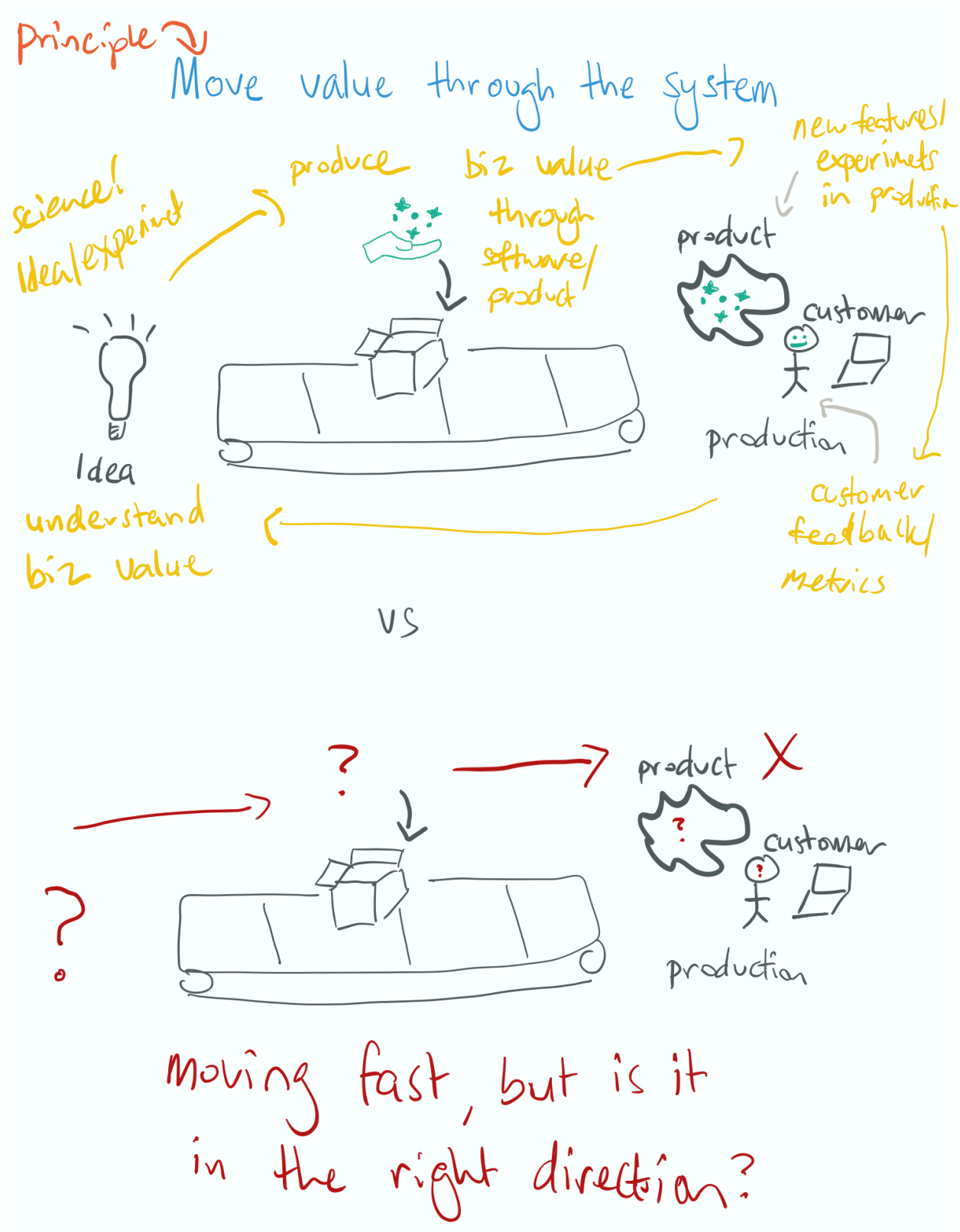
# Transformation on a page:

## Grouping themes to show how they build on & depend on each other

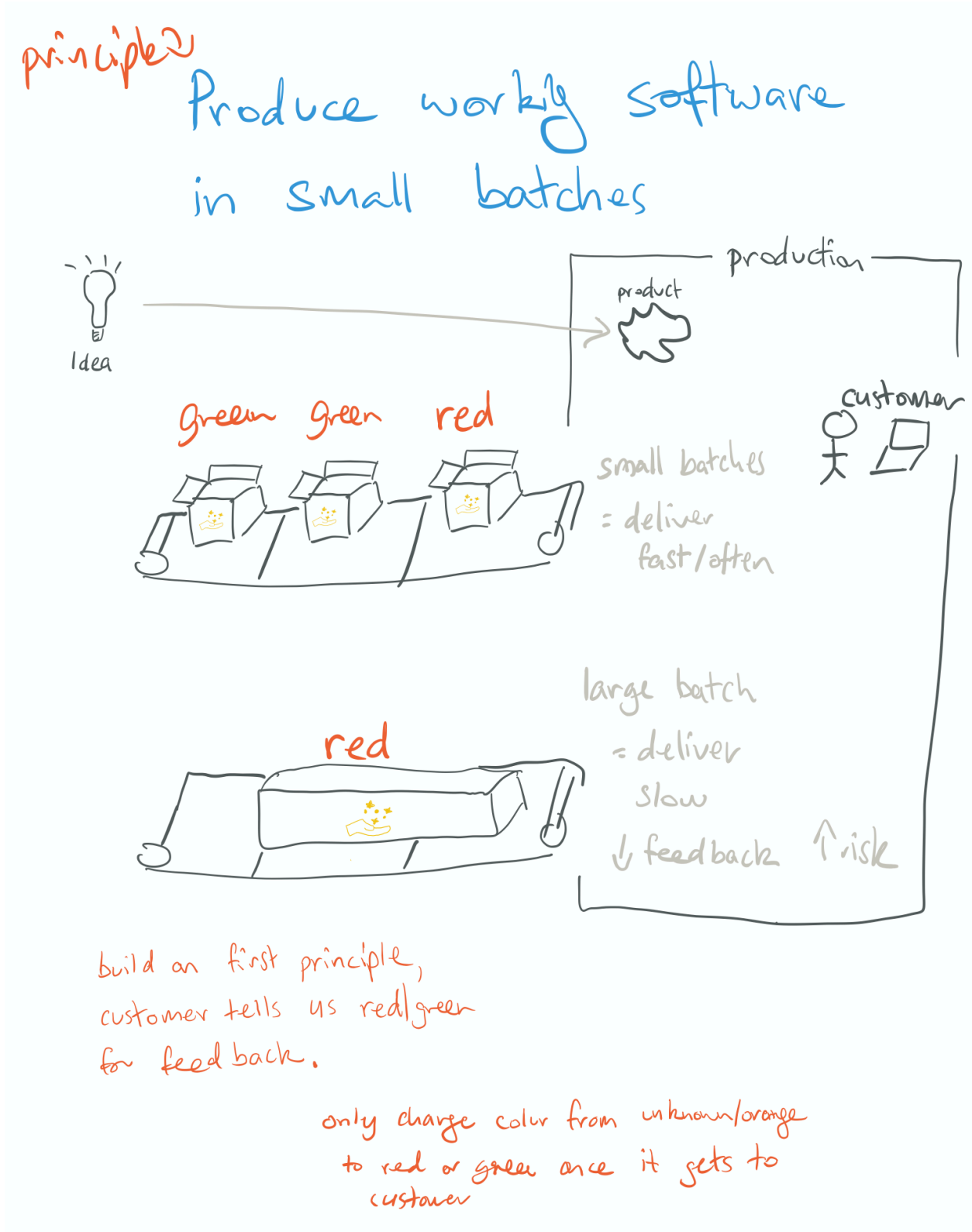



Product teams
Understand business value
short feedback loop w/ customer
VALUE STREAM MAPPING
Agile/Scrum/kanban/XP
Don't be overly prescriptive
Understand why you're doing what you're doing
lean
A learning org
Growth Mindset
Encourage innovation & collaboration
blameless postmortems
embrace open source
Improved collab cross-dept
Continuous Improvement
Demo Days, Office Hours, Slack
Gamedays
Empowered teams
No key person deps.
Social Contracts, High Performing teams (trust)
people influences tech
development maturity
Maturity Assessments
Capture metrics
Display metrics in real-time (dashboards/TVs)
APIs & status page for all your software
SRE
development uplift
Trunk based Development
Test first, shift left some more
Run your pipeline tests before you commit
technical enablement for development
Cloud Account Provisioning w/ Guardrails
Pipeline templates w/ checks, but full access to modify by dev
Self Service Development Full Access in Dev
Prod-like data available in dev

Move value through the system:



Work in small batches:




BTW working software means:

# Workly software means...