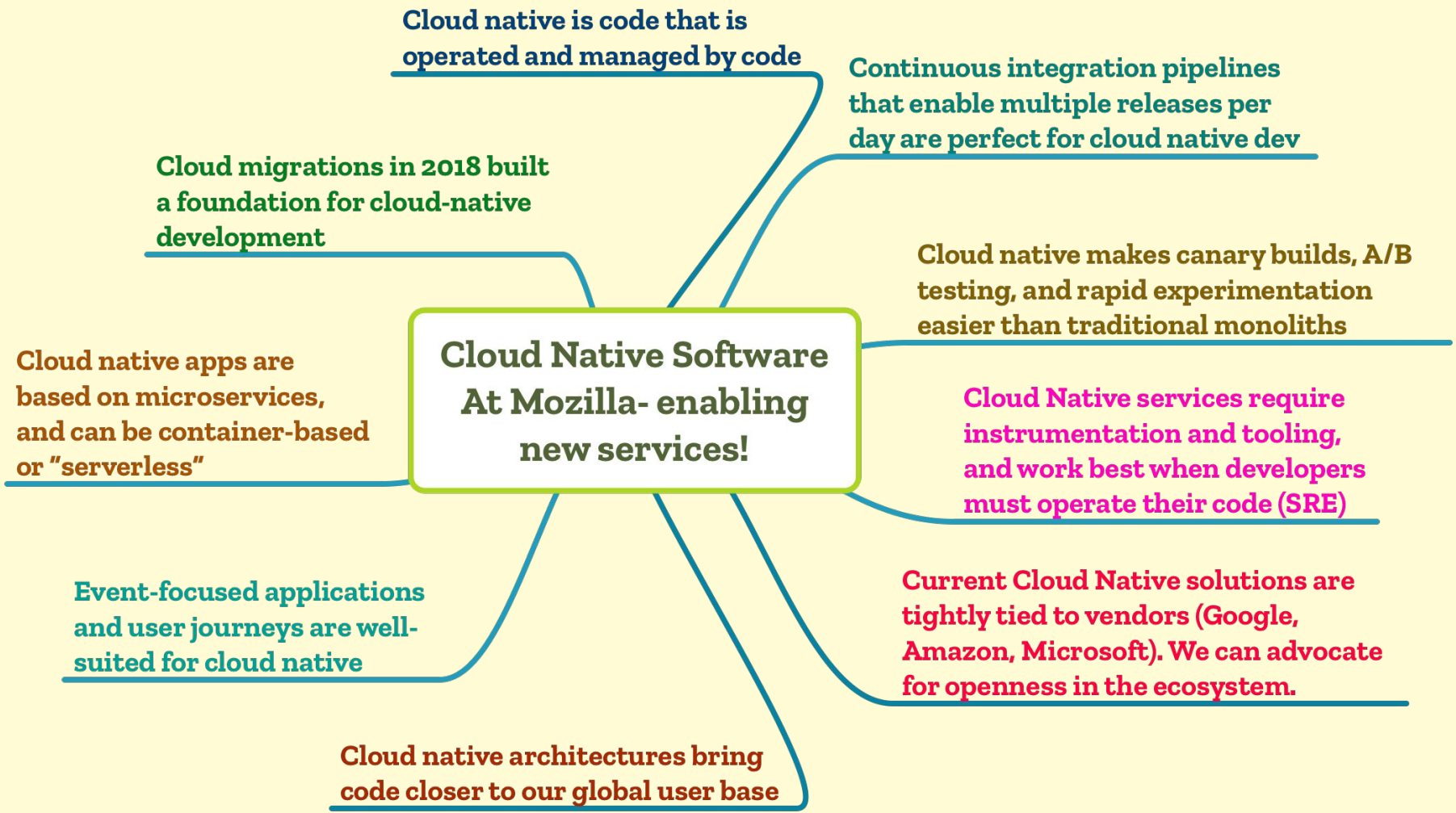
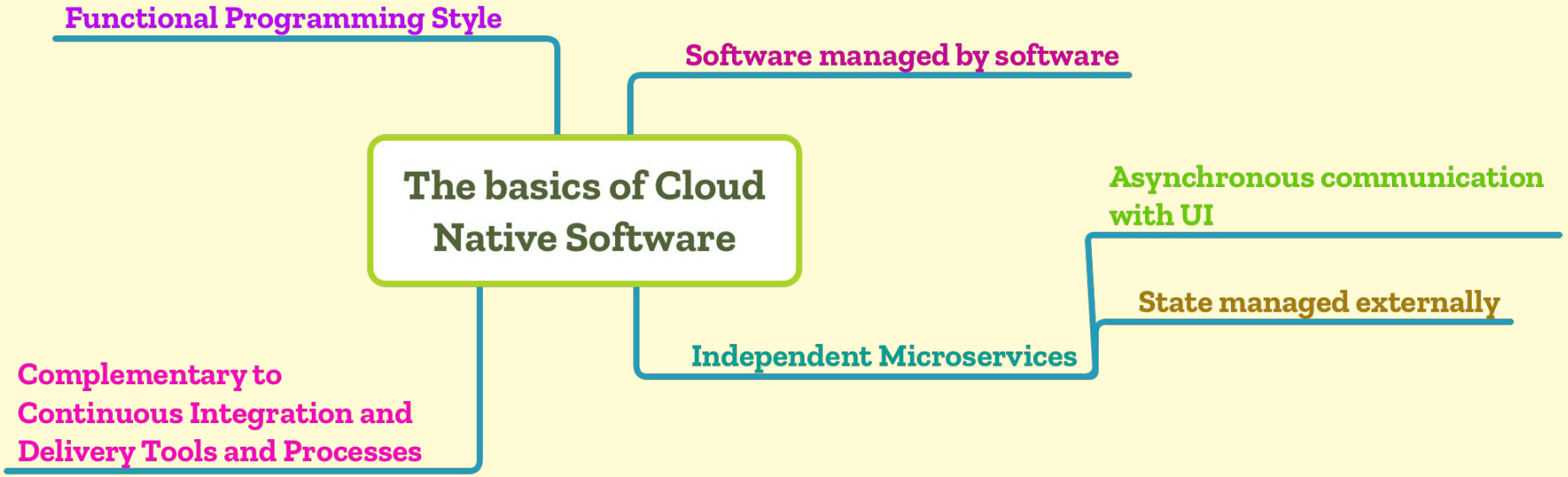
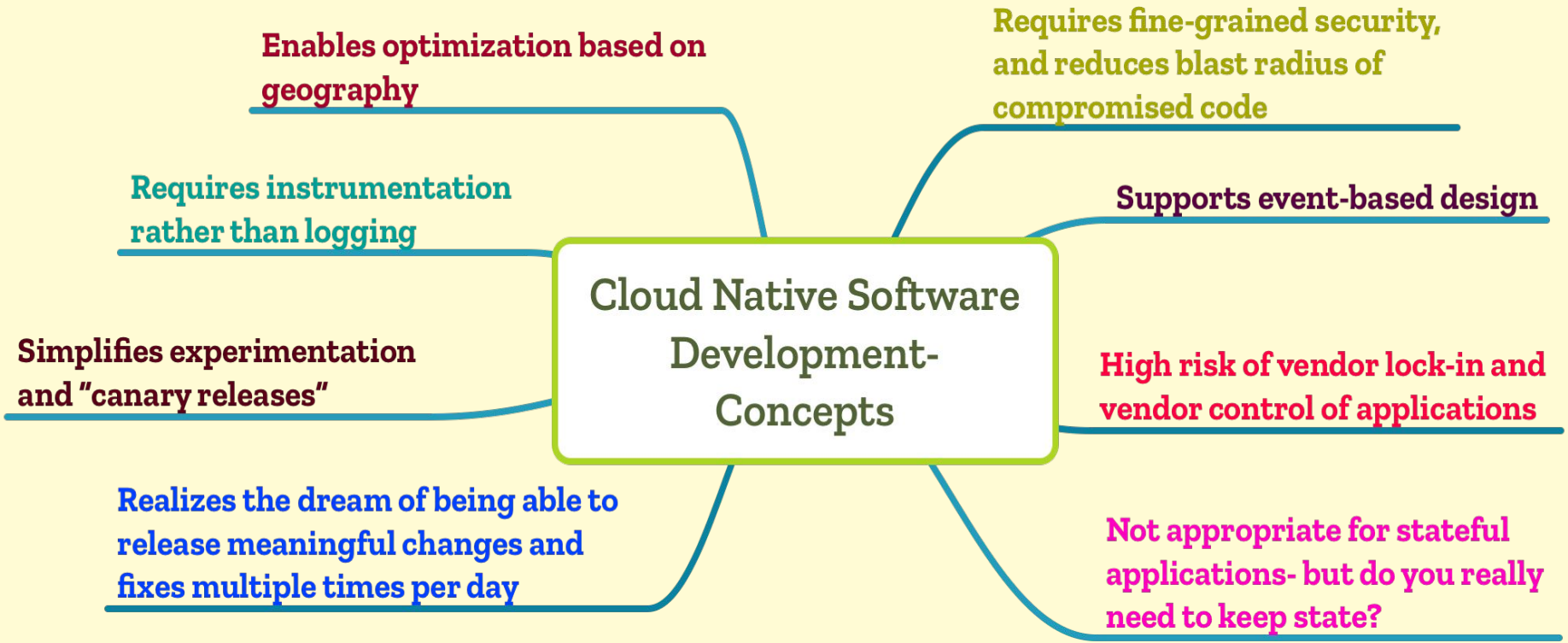
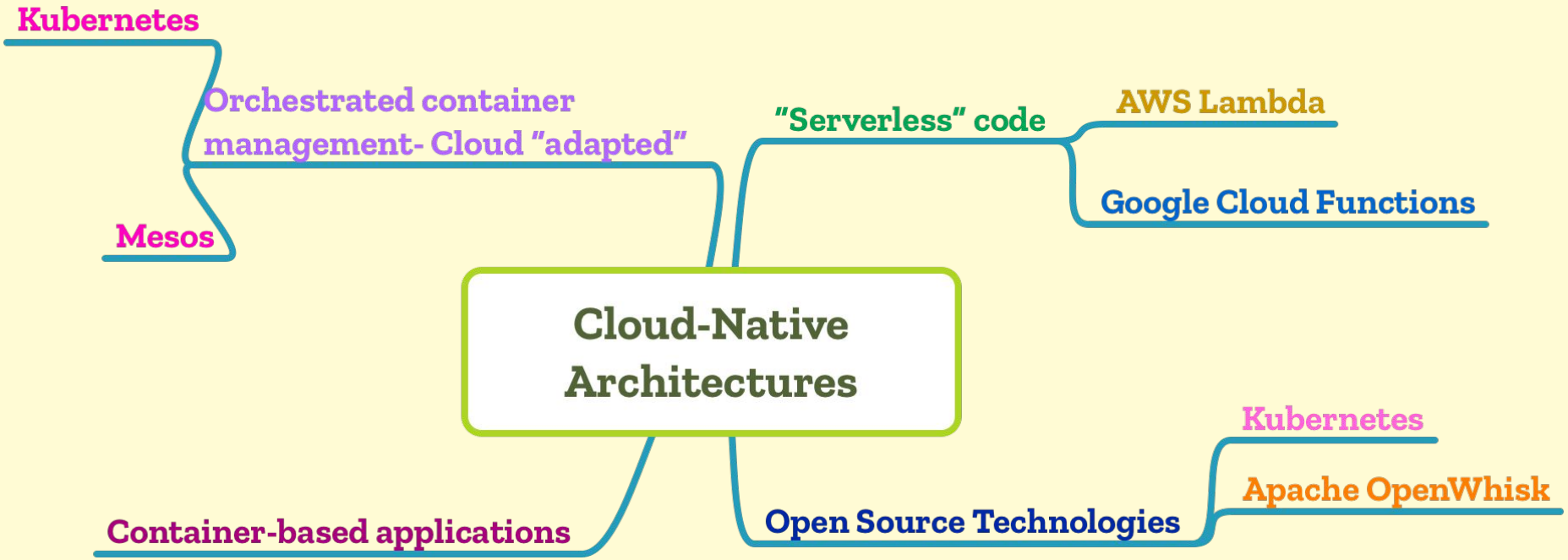
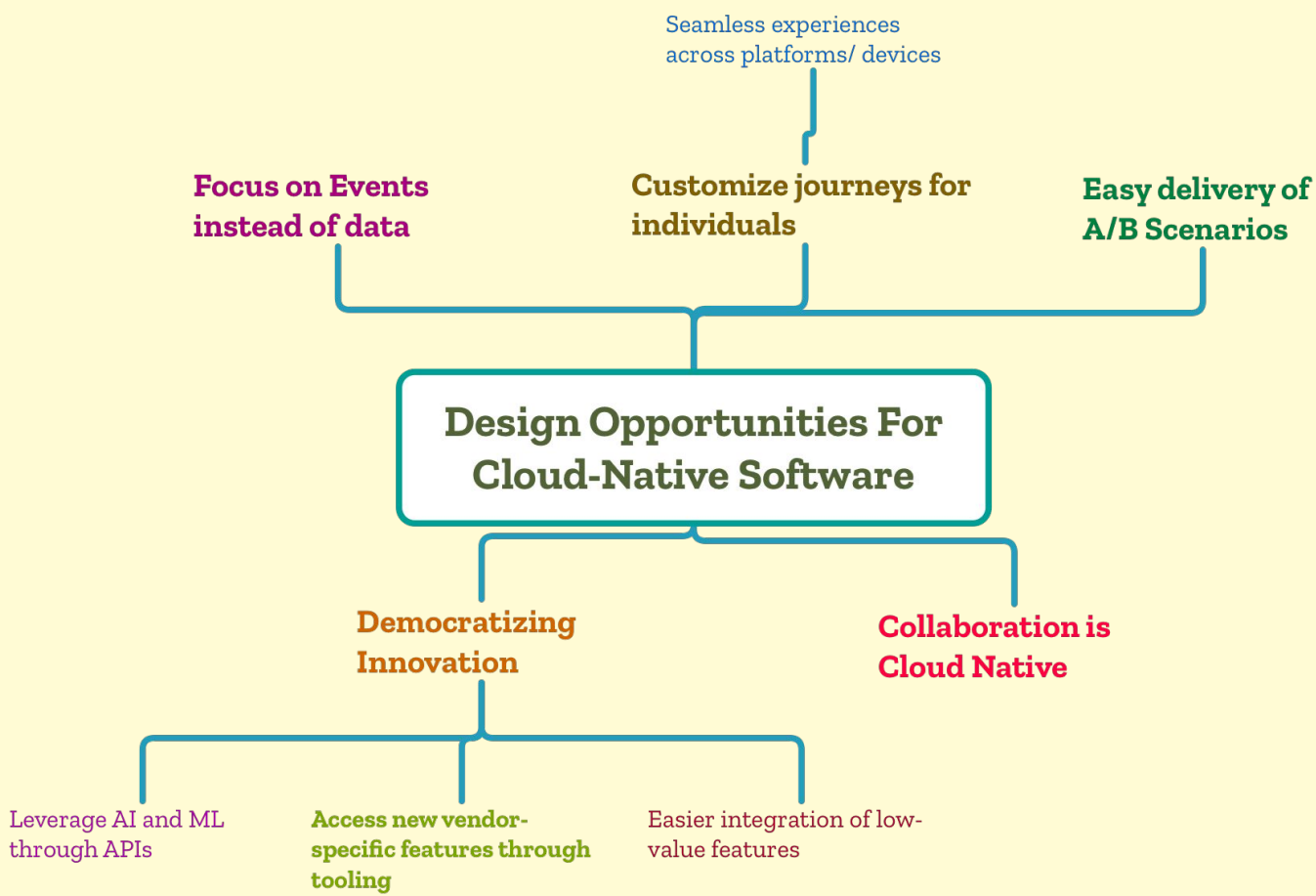

Cloud native is code that is operated and managed by code
Continuous integration pipelines that enable multiple releases per day are perfect for cloud native dev
Cloud migrations in 2018 built a foundation for cloud-native development
Cloud native makes canary builds, A/B testing, and rapid experimentation easier than traditional monoliths
Cloud Native Software At Mozilla- enabling new services!
Cloud native apps are based on microservices, and can be container-based or "serverless"
Cloud Native services require instrumentation and tooling, and work best when developers must operate their code (SRE)
Event-focused applications and user journeys are well-suited for cloud native
Current Cloud Native solutions are tightly tied to vendors (Google, Amazon, Microsoft). We can advocate for openness in the ecosystem.
Cloud native architectures bring code closer to our global user base


Functional Programming Style
Software managed by software
The basics of Cloud Native Software
Asynchronous communication with UI
State managed externally
Independent Microservices
Complementary to Continuous Integration and Delivery Tools and Processes

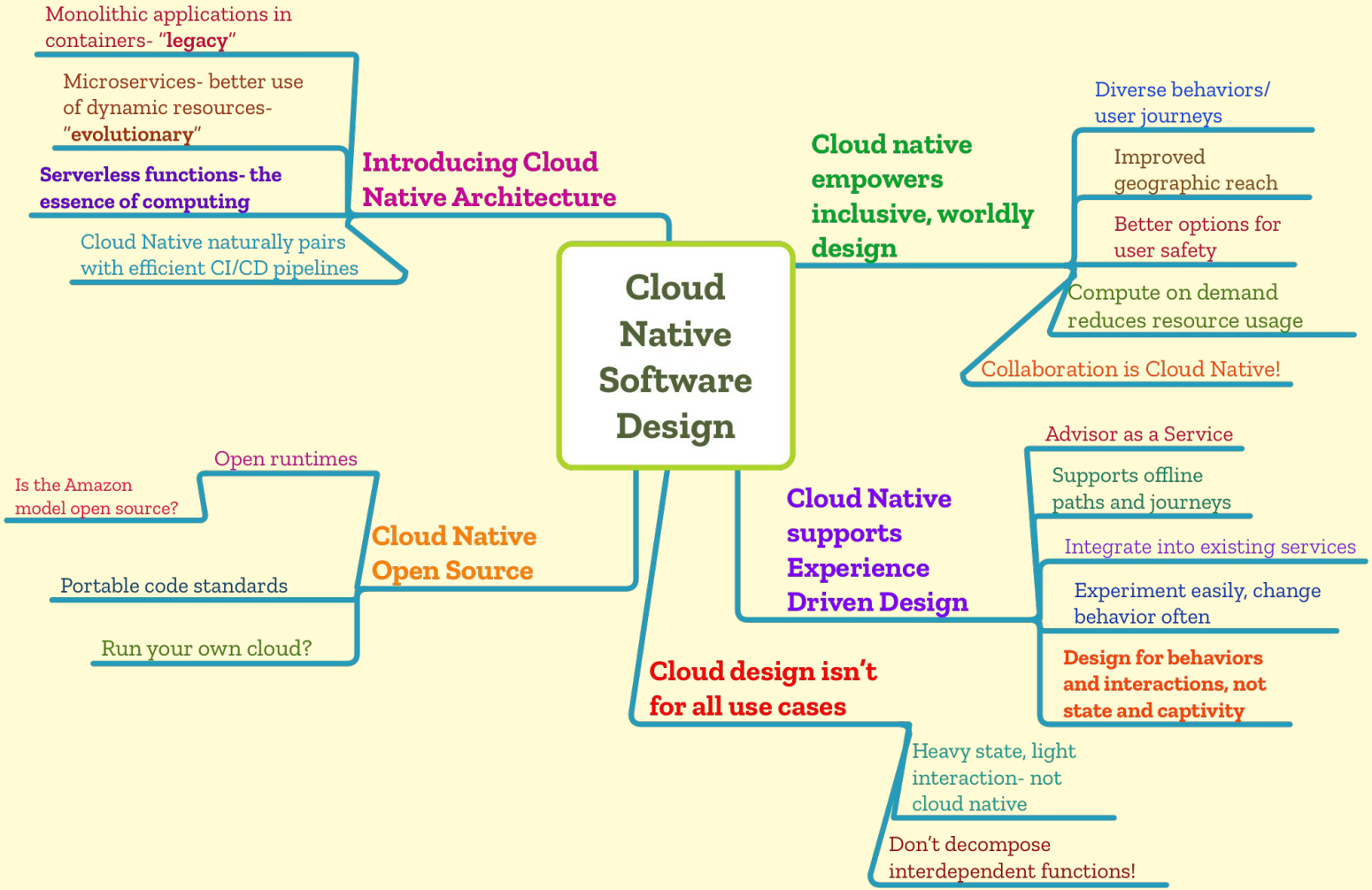

Cloud Native Software Development- Concepts
Enables optimization based on geography
Requires instrumentation rather than logging
Simplifies experimentation and "canary releases"
Realizes the dream of being able to release meaningful changes and fixes multiple times per day
Requires fine-grained security, and reduces blast radius of compromised code
Supports event-based design
High risk of vendor lock-in and vendor control of applications
Not appropriate for stateful applications- but do you really need to keep state?

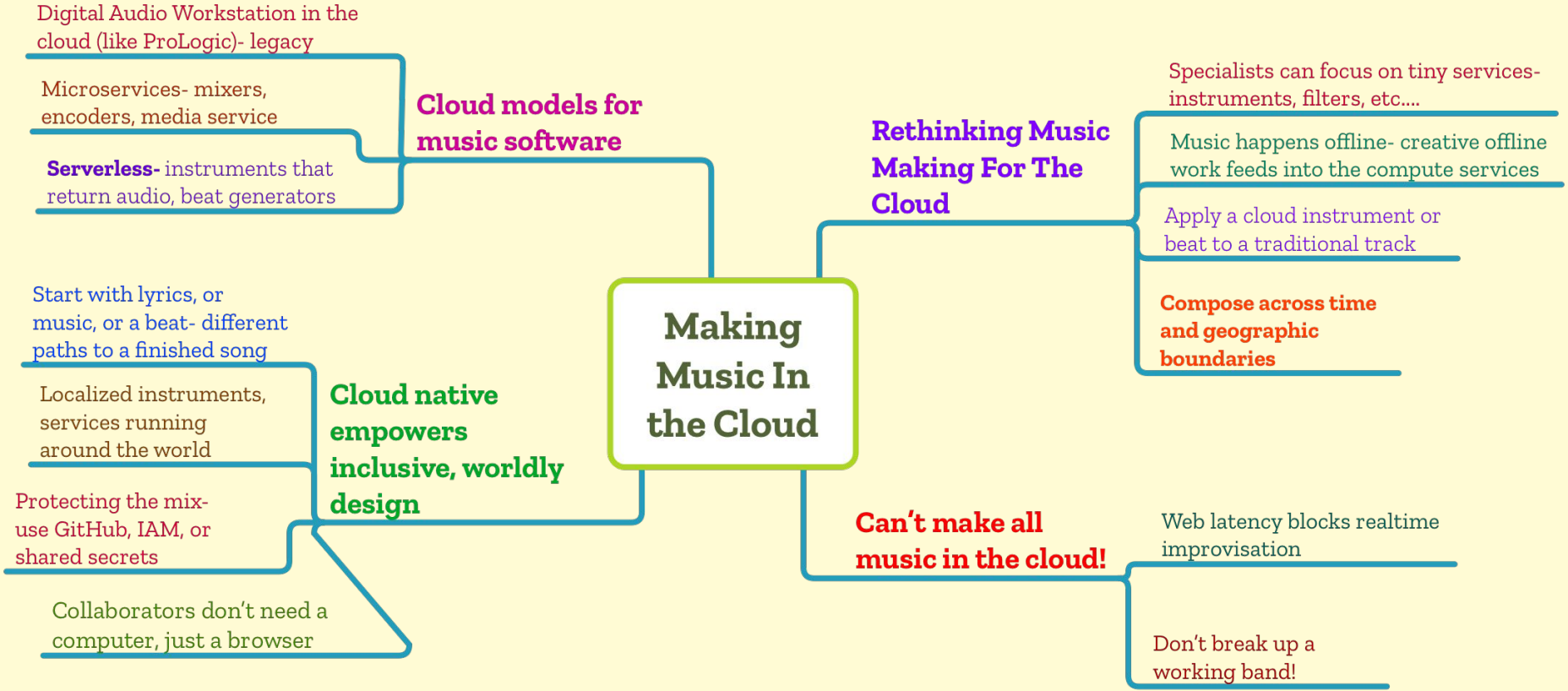

Kubernetes
Orchestrated container management- Cloud "adapted"
Mesos
"Serverless" code
AWS Lambda
Google Cloud Functions
Cloud-Native Architectures
Container-based applications
Open Source Technologies
Kubernetes
Apache OpenWhisk



see https://www.uxmatters.com/mt/archives/2018/02/the-opportunities-of-cloud-computing.php

Patterns/ Anti-patterns for Cloud Native software


Cloud Native Software Design
Introducing Cloud Native Architecture
Monolithic applications in containers- "legacy"
Microservices- better use of dynamic resources- "evolutionary"
Serverless functions- the essence of computing
Cloud Native naturally pairs with efficient CI/CD pipelines
Cloud native empowers inclusive, worldly design
Diverse behaviors/ user journeys
Improved geographic reach
Better options for user safety
Compute on demand reduces resource usage
Collaboration is Cloud Native!
Cloud Native Open Source
Open runtimes
Is the Amazon model open source?
Portable code standards
Run your own cloud?
Cloud Native supports Experience Driven Design
Advisor as a Service
Supports offline paths and journeys
Integrate into existing services
Experiment easily, change behavior often
Design for behaviors and interactions, not state and captivity
Cloud design isn't for all use cases
Heavy state, light interaction- not cloud native
Don't decompose interdependent functions!


Making Music In the Cloud
Cloud models for music software
Digital Audio Workstation in the cloud (like ProLogic)- legacy
Microservices- mixers, encoders, media service
Serverless- instruments that return audio, beat generators
Rethinking Music Making For The Cloud
Specialists can focus on tiny services- instruments, filters, etc....
Music happens offline- creative offline work feeds into the compute services
Apply a cloud instrument or beat to a traditional track
Compose across time and geographic boundaries
Cloud native empowers inclusive, worldly design
Start with lyrics, or music, or a beat- different paths to a finished song
Localized instruments, services running around the world
Protecting the mix- use GitHub, IAM, or shared secrets
Collaborators don't need a computer, just a browser
Can't make all music in the cloud!
Web latency blocks realtime improvisation
Don't break up a working band!