LUNA VISION

OUR DESIGN PRINCIPLES

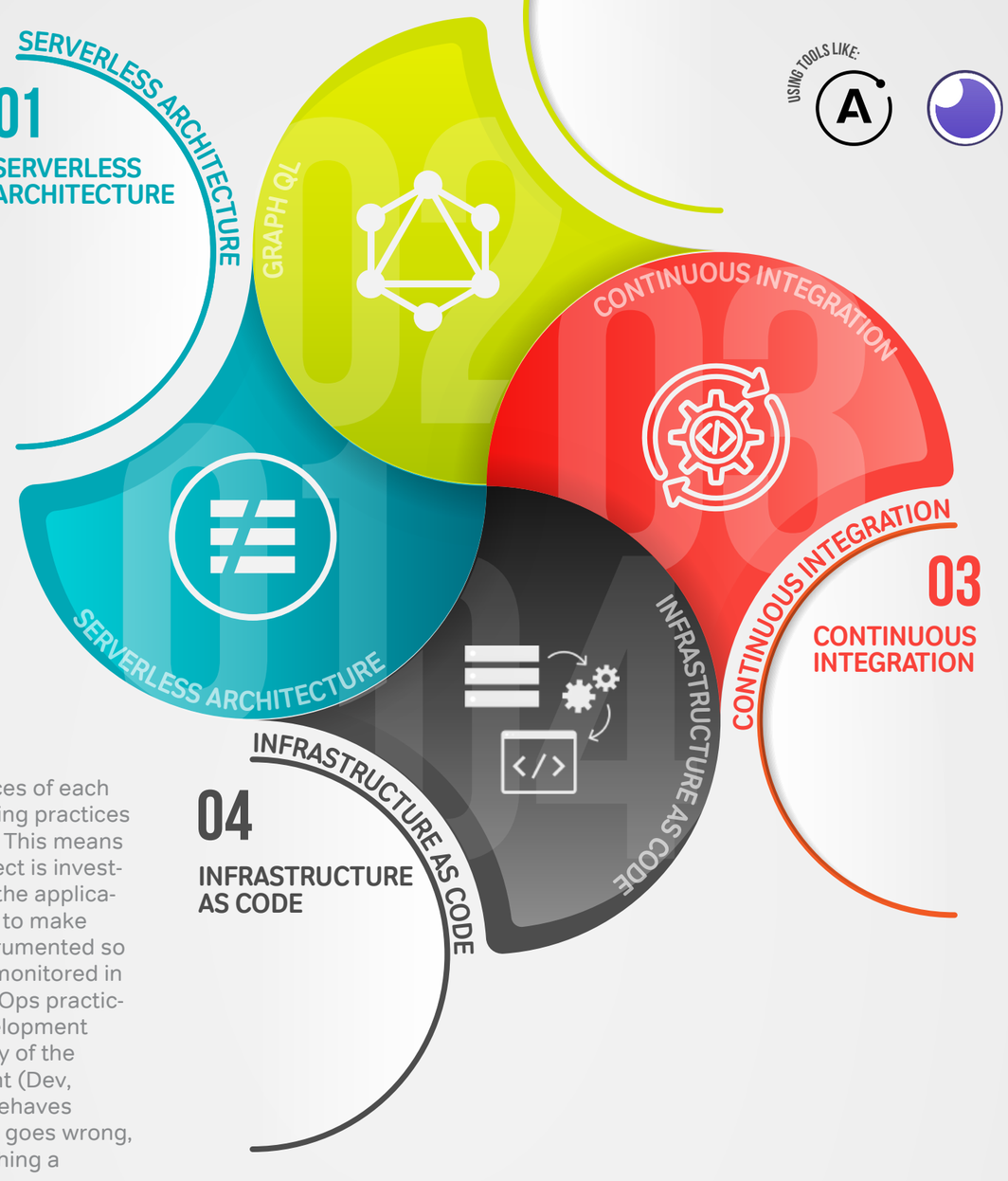
Lineate LunaVision™ is the technical approach we use to drive quick, clean, future-proof applications. Our focus is nimble application development that both exemplifies and drives modern software development practices. The goal is to get business functionality up quickly, easily scale up to billions of transactions if necessary, and support rapid evolution without incurring technical debt.

Applications should be simple to change, predictable, and always on.

4 PILLARS OF LUNAVISION

[**WHAT** WE BUILD]

how we maintain modern and flexible deployment practices that easily evolve over time.



01 SERVERLESS ARCHITECTURE

USING TOOLS LIKE:

Offload as much utility-type engineering and provisioning of resources as possible. Leverage infrastructure-as-a-service providers such as Amazon Web Services and Google Cloud who have perfected maintaining these with minimal risk. As much of your software investment as possible should go towards directly improving your core business.

02 GRAPHQL

USING TOOLS LIKE:

Use application integration technologies such as GraphQL (one of many) to promote discoverability. Minimize dependencies between teams, and empower them to deploy frequently on their own.

03 CONTINUOUS INTEGRATION

USING TOOLS LIKE:

Design and implement Continuous integration tests as the application is coded, and incorporate them to automatically run during the build process. If the tests don't pass, the application isn't deployed. Make this a non-negotiable part of the application development lifecycle. We have found that each hour invested up front pays itself back many times over by the time the functionality goes live.

04 INFRASTRUCTURE AS CODE

USING TOOLS LIKE:

Provision the servers and services of each application using the same coding practices as the rest of the functionality. This means that every engineer on the project is invested in making it trivial to deploy the application without complications, and to make sure the application is fully instrumented so its health and behavior can be monitored in real time. By driving these DevOps practices to every member of the development team, we ensure that every copy of the application in every environment (Dev, Staging, Production, or other) behaves exactly the same. If something goes wrong, redeploying is as simple as pushing a button.

TWIN PRECEPTS

[**HOW** WE BUILD]

how we ensure every application has a trivial path to scale with the business, and that the applications behave with certainty under the inherently uncertain conditions of the real world.

01 SCALE HORIZONTALLY

Every component should behave predictably regardless of the number of copies running, the number of servers used, the time it is invoked, or the conditions under which it was invoked.

This doesn't mean that simply throwing more hardware at a problem makes things better, but it does provide a clear and standard way to grow applications when needed.

02 DESIGN FOR FAILURE

Every component should be written with the expectation that things can fail at any moment. Restarting should have no side-effects. Partial states are understood and handled.

Crucially this means that restarting a service will pick up where it left off without side-effects such as double billing. Combined with the Scale Horizontally precept, it means each copy of a service can seamlessly take over from one that failed.

LEARN MORE ABOUT WHAT WE BUILD AND HOW WE BUILD IT:

OUR MANIFESTO | CASE STUDIES | LUNAVISION HACKATHON

TALK TO US ABOUT WHAT YOU'RE BUILDING:

ADVICE FROM OUR SA TEAM | KICKOFF A 2-WEEK D/A | GET A PROPOSAL

Lineate