Configure Once, Run Everywhere

How and Why to Use a Common Configuration for Dev, Testing, and CI Environments
About Garden

- Founded in 2018
- Headquartered in Berlin, Germany (with users and customers around the globe)
- We provide an automation platform for cloud native development and testing
- Open Core
My dislikes

- Frequent context switching
- Repetition
- Vagueness
My likes

- Flow state
- Smooooth automation
- Clarity and sincerity
- 80s Japanese synth-pop
What makes the current dev/CI/CD experience less than ideal?
Problems with the traditional dev process

- devs + QA
- review
- local env
- build & test
- merge
- staging / integration
- promote
- production

inner loop / local

CI / CD
Problems with the traditional dev process

Local dev/preview environments are difficult (or not even feasible) to set up and maintain.

Integration and end-to-end tests are difficult to achieve, and feedback comes very late in the process.
Problems with the traditional dev process

Debugging failures in CI is incredibly tedious and slow

A lot of discrepancies between tooling, configuration and workflows for each phase in the dev/delivery cycle.
The Garden approach:
Configure Once, Run Anywhere
The Stack Graph

- Codify all the build and runtime dependencies across your project.
- Garden uses this to **only build, deploy and test what's affected by your changes**—so that devs don't need to wait for full pipeline runs for every little change.
Simple, distributed configuration

- Each component *describes itself*—It's a distributed system, so your configuration shouldn't be monolithic.
- Powerful templating syntax.
- Define your own abstractions through custom templates.
Same setup for dev and CI

- Run `garden deploy` or `garden test` — the same during development and in CI
- Use the same cluster for both, and get a **shared cache** between all your developers and pipelines
- Waste less time debugging CI
What this gives us

- The developer gets a preview env and proper test feedback even before pushing to CI.
- In CI we use the same tools and configuration as for inner-loop development.

Diagram:
1. Dev
2. Code
3. Unit tests
4. Dev preview
5. Integ/E2E tests
6. Pull request (PR)
   - Build & test
   - Integ + E2E
   - Preview env
7. Merge
   - Production

Garden

Inner loop / Local

CI / CD
A long list of benefits

- Production-like environments at every stage, on-demand
- Run all or individual integration tests at any stage, even before pushing to git.
  - Much faster to write and debug integ/e2e tests!
- No more pushing "asdfasdfs" commits to CI for debugging
- Your build, test and deploy configuration is portable across CI systems
- Share build and test caches between developers and across dev + CI
- Stay focused, stay in flow
Wrapping up

- To get started: docs.garden.io
- We'd love to hear what you think! Please ask us questions along the way: community.garden.io
- We're hiring: garden.io/careers
- jon@garden.io / @JonEdvald
Thank you!