Built-in Application Resiliency

Allan Shone, Senior Systems Engineer at Deputy.

@CerealBoy

Being Resilient

What does it mean to be resilient?

- Being fault tolerant.

Being Resilient

What does it mean to be resilient?

- Being fault tolerant.
- Exhibiting consistent behaviour.

Being Resilient

What does it mean to be resilient?

- Being fault tolerant.
- Exhibiting consistent behaviour.
- Recovering from failure.

Being Resilient

What does it mean to be resilient?

- Being fault tolerant.
- Exhibiting consistent behaviour.
- Recovering from failure.
- Comprehensively observable.

Being Resilient

What does it mean to be resilient?

- Being fault tolerant.
- Exhibiting consistent behaviour.
- Recovering from failure.
- Comprehensively observable.
- Planned recovery from disasters.

Being Resilient

What does it mean to be resilient?

- Being fault tolerant.
- Exhibiting consistent behaviour.
- Recovering from failure.
- Comprehensively observable.
- Planned recovery from disasters.
- Adapting to change.

Recovering from Disaster

Disaster Recovery

Preparing for the worst.

- Solid foundation to build upon.

Disaster Recovery

Preparing for the worst.

- Solid foundation to build upon.
- Ability to get back to basics when the unspeakable happens.

Disaster Recovery

Preparing for the worst.

- Solid foundation to build upon.
- Ability to get back to basics when the unspeakable happens.
- Heavy use of infrastructure as code.

Disaster Recovery

Preparing for the worst.

- Solid foundation to build upon.
- Ability to get back to basics when the unspeakable happens.
- Heavy use of infrastructure as code.
- Reproducible processes for all the details.

Disaster Recovery

Preparing for the worst.

- Solid foundation to build upon.
- Ability to get back to basics when the unspeakable happens.
- Heavy use of infrastructure as code.
- Reproducible processes for all the details.
- Provides scaling capabilities.

Moving Quickly

- Starting from disaster recovery allows for rapid growth.

Moving Quickly

- Starting from disaster recovery allows for rapid growth.
- Fresh eyes onboard smoother.

Moving Quickly

- Starting from disaster recovery allows for rapid growth.
- Fresh eyes onboard smoother.
- Full system replication.

Moving Quickly

- Starting from disaster recovery allows for rapid growth.
- Fresh eyes onboard smoother.
- Full system replication.
- Component changes comprehensively testable.

Resilience in Disaster

- Starting fresh can be an option, if it’s available.

Resilience in Disaster

- Starting fresh can be an option, if it’s available.
- Using the plan to prepare as disaster looms can prevent loss.

Resilience in Disaster

- Starting fresh can be an option, if it’s available.
- Using the plan to prepare as disaster looms can prevent loss.
- With disaster as the guide, preparedness can be better achieved.

Managing Change

- Deployments can be a great facilitator.

Managing Change

- Deployments can be a great facilitator.
- Minimising time to change can be a big factor in maintaining resiliency.

Fault Tolerance

Being Tolerant

Gracefully handling downstream issues.

- Use patterns that enable this handling, such as circuit breakers.

Being Tolerant

Gracefully handling downstream issues.

- Use patterns that enable this handling, such as circuit breakers.
- Be mindful of usage patterns.

Being Tolerant

Gracefully handling downstream issues.

- Use patterns that enable this handling, such as circuit breakers.
- Be mindful of usage patterns.
- Respond appropriately for behaviour expectations.

Orchestration Help

Without changing the application itself?

- Using a service mesh.

Orchestration Help

Without changing the application itself?

- Using a service mesh.
- Using a load balancer.

Orchestration Help

Without changing the application itself?

- Using a service mesh.
- Using a load balancer.
- Using DNS.

Centralise Executions

- Internal libraries ease the implementation.

Centralise Executions

- Internal libraries ease the implementation.
- Set the right defaults that are tolerant within the system.

Centralise Executions

- Internal libraries ease the implementation.
- Set the right defaults that are tolerant within the system.
- Defining expectations on all components.

Centralise Executions

- Internal libraries ease the implementation.
- Set the right defaults that are tolerant within the system.
- Defining expectations on all components.
- Trade-off the edge-case with a general approach.

Centralise Executions

- Internal libraries ease the implementation.
- Set the right defaults that are tolerant within the system.
- Defining expectations on all components.
- Trade-off the edge-case with a general approach.
- Good place for other patterns, like exponential backoff.

Recovering from Failure

Failure Happens

What to do about it?

- Again, set expectations.

Failure Happens

What to do about it?

- Again, set expectations.
- Internal services have the benefit of control, define and manage it.

Failure Happens

What to do about it?

- Again, set expectations.
- Internal services have the benefit of control, define and manage it.
- Buffers and caches can help.

Failure Happens

What to do about it?

- Again, set expectations.
- Internal services have the benefit of control, define and manage it.
- Buffers and caches can help.
- Be graceful with degradation.

Buffering

- Video is the prime example of this.

Buffering

- Video is the prime example of this.
- Validate the internal buffer system.

Buffering

- Video is the prime example of this.
- Validate the internal buffer system.
- Consider a longer term solution, especially in the event of recurrence.

Buffering

- Video is the prime example of this.
- Validate the internal buffer system.
- Consider a longer term solution, especially in the event of recurrence.
- Push things out of band where possible.

Out of band

- Catch up quicker, when there's no traffic to worry about.

Out of band

- Catch up quicker, when there's no traffic to worry about.
- Services continue to utilise the buffer, helping them to continue to be high performing.

Out of band

- Catch up quicker, when there's no traffic to worry about.
- Services continue to utilise the buffer, helping them to continue to be high performing.
- Entire change of architecture.

Out of band

- Catch up quicker, when there's no traffic to worry about.
- Services continue to utilise the buffer, helping them to continue to be high performing.
- Entire change of architecture.
- Be mindful of things like duplication, however.

Behaving Consistently

Consistency

- Being consistent drives home the premise of reliability.

Consistency

- Being consistent drives home the premise of reliability.
- Resiliency is born out of the ability to remain consistent while under duress.

Consistency

- Being consistent drives home the premise of reliability.
- Resiliency is born out of the ability to remain consistent while under duress.
- Using a buffer is a great way to exhibit this.

Adaptation

When drastic measures are needed.

- Add on to a behaviour when required rather than change it.

Being Observable

Visibility

Proving the resiliency.

- Ensuring that everything is behaving as expected.

Visibility

Proving the resiliency.

- Ensuring that everything is behaving as expected.
- Generate baselines that can be extrapolated from.

Visibility

Proving the resiliency.

- Ensuring that everything is behaving as expected.
- Generate baselines that can be extrapolated from.
- Gain an understanding of general reliability.

Feedback Loop

Self reliance.

- With data made available, systems can adapt.

Feedback Loop

Self reliance.

- With data made available, systems can adapt.
- Intelligent execution of required actions.

Resiliency

To Being Resilient!

- Manage expectations.

To Being Resilient!

- Manage expectations.
- Be graceful.

To Being Resilient!

- Manage expectations.
- Be graceful.
- Have the ability to move quickly.

To Being Resilient!

- Manage expectations.
- Be graceful.
- Have the ability to move quickly.
- Adapt as required.

Thank you!

@CerealBoy