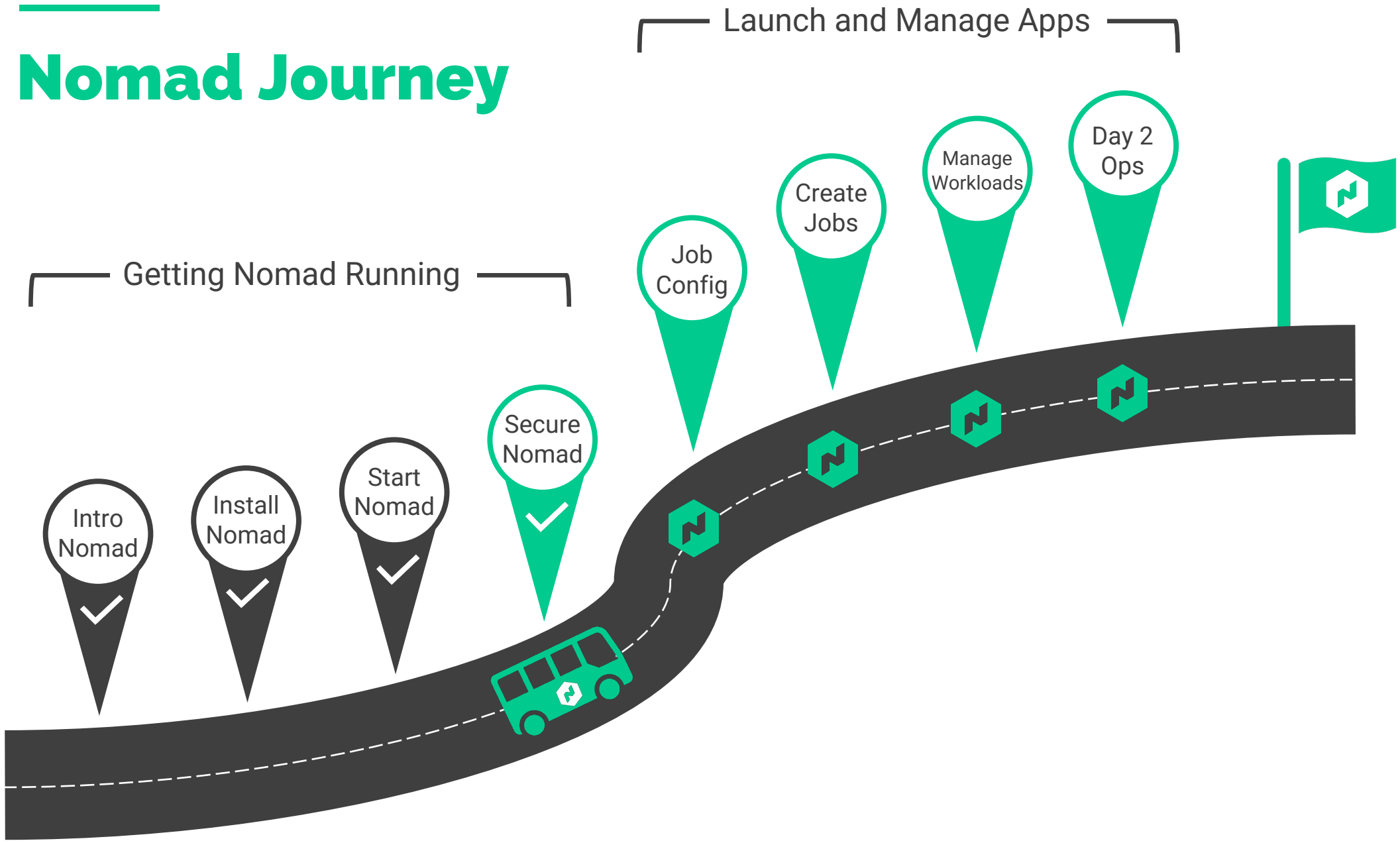
Working with
Nomad Jobs

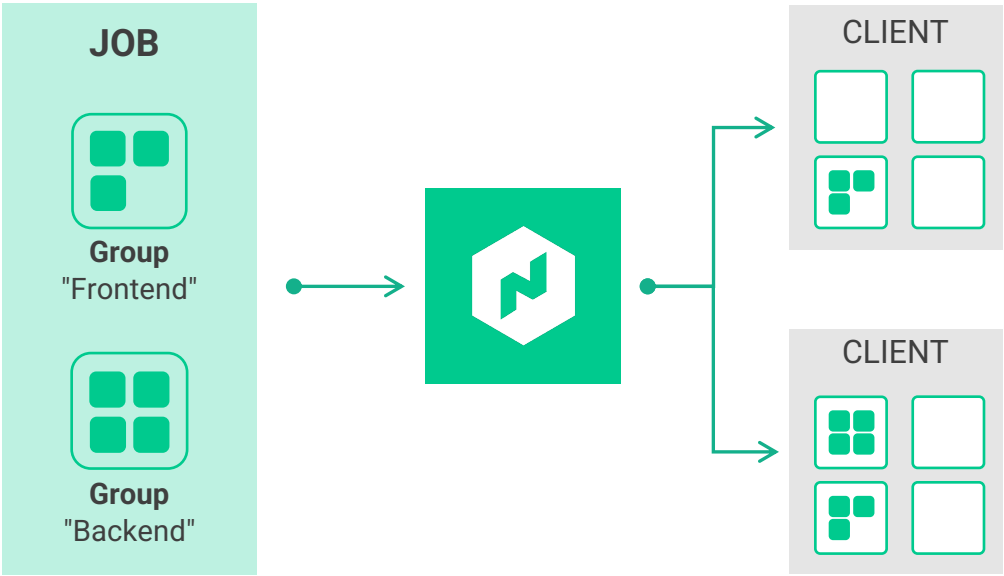
Nomad Journey



Core Concepts

(Reminder)

Task	The smallest unit of scheduling work. It could be a Docker container, a Java application, or batch processing.
Group	A series of tasks that should be co-located on the same Nomad client. Tightly-coupled tasks in the same group can share the same network/storage
Job	The declarative that defines the deployment rules for applications
Application	The instance of a task group that are running on client



I'm Ready....How Do We Run a Job?

First, we need to write our Job spec

- Each job is submitted to Nomad using a job specification (job spec)
- The job file is written in HCL (or JSON) and is often saved with a `.nomad` extension
- Job files only contain one job, but they can have multiple tasks & groups if needed
- Tasks define the actual work that will be executed while the driver controls how the task is executed

Job Specification Hierarchy

- job
  # ...
  - group
    # ...
    - task
      # ...
      - resources
        # ...
      - constraint
        # ...
    - task

GOAL:
Start with a small job spec and build up from there

Creating a Nomad Job Specification

Nomad Job Specification

job

TERMINAL

```
job "tetris" {
  # ...
}
```

Nomad Job Specification

job

TERMINAL

```
job "tetris" {
  # ...

  # Specify the datacenters this job can run in
  datacenters = ["dc1"]
  # ...
}
```

Nomad Job Specification

job

TERMINAL

```
job "tetris" {
  # ...

  # Specify the datacenters this job can run in
  datacenters = ["dc1"]
  type = "service"
  # ...
}
```

Nomad Job Specification

job

TERMINAL

```
job "tetris" {
  # ...

  datacenters = ["dc1"]
  type = "service"
  constraint {
    attribute = "${attr.kernel.name}"
    value     = "linux"
  }
  # ...
}
```

Nomad Job Specification

job

TERMINAL

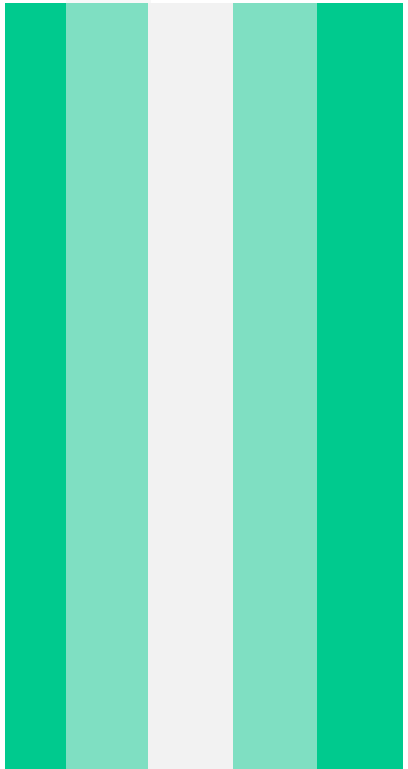
```
job "tetris" {
  # ...

  datacenters = ["dc1"]
  type = "service"
  constraint {
    attribute = "${attr.kernel.name}"
    value     = "linux"
  }
  update {
    max_parallel = 1
  }
  # ...
}
```

Nomad Job Specification

TERMINAL

```
job "tetris" {
  # ...

  datacenters = ["dc1"]

  group "games" {
    count = 1
    # ...
  }

  # ...
}
```

group

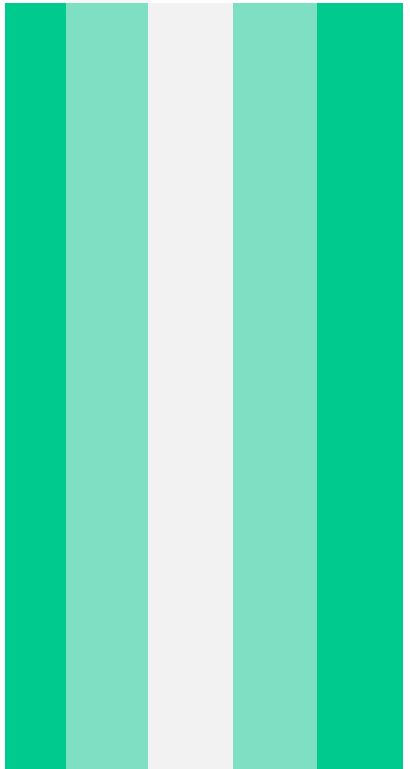


Nomad Job Specification

TERMINAL

```
job "tetris" {
  # ...

  datacenters = ["dc1"]

  group "games" {
    count = 1

    task "tetris" {
      driver = "docker"

      config {
        image           = "bsord/tetris"
        ports           = ["web"]
        auth_soft_fail = true
      }
    }
  }
}
```

group



Nomad Job Specification

TERMINAL

```
job "tetris" {
  # ...

  group "games" {
    count = 1

    task "tetris" {
      driver = "docker"

      config {
        image = "bsord/tetris"
        ports = ["web"]
        auth_soft_fail = true
      }
      resources {
        cpu    = 500 # 500MHz
        memory = 256 # 256MB
        network {
          mbits = 10
        }
      }
    }
```

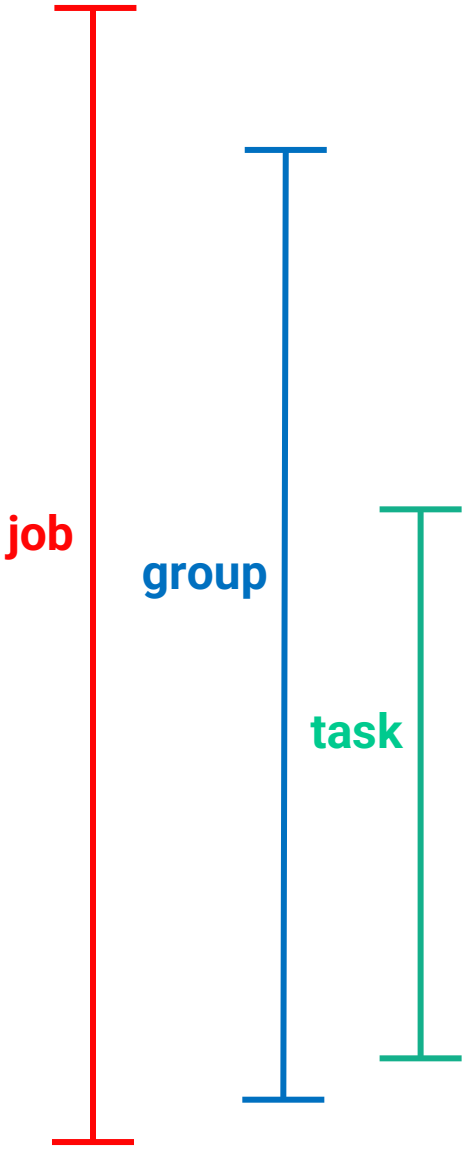
Please assume I closed all my brackets here ☺

group

Complete File

With a Few Additions

- All these configurations are in a single `.nomad` file
- The file can be stored in a code repo and iterated on as needed
- The file will be submitted to Nomad to create our resources when we're ready to launch our application




```
job "tetris" {
  datacenters = ["dc1"]

  group "games" {
    count = 1

    network {
      port "web" {
        to = 80
      }
    }

    task "tetris" {
      driver = "docker"

      config {
        image          = "bsord/tetris"
        ports          = ["web"]
        auth_soft_fail = true
      }

      resources {
        cpu    = 50
        memory = 256
      }
    }
  }
}
```

Validate Job Spec

- Nomad CLI supports multiple ways to validate and format your job specification file
- Use `nomad fmt` to format the job spec file to a canonical format
  - This will automatically format all `.nomad` or `.hcl` files in the directory where the command is run
- Use `nomad validate <file>` to check a job spec for any syntax errors or validation problems

TERMINAL

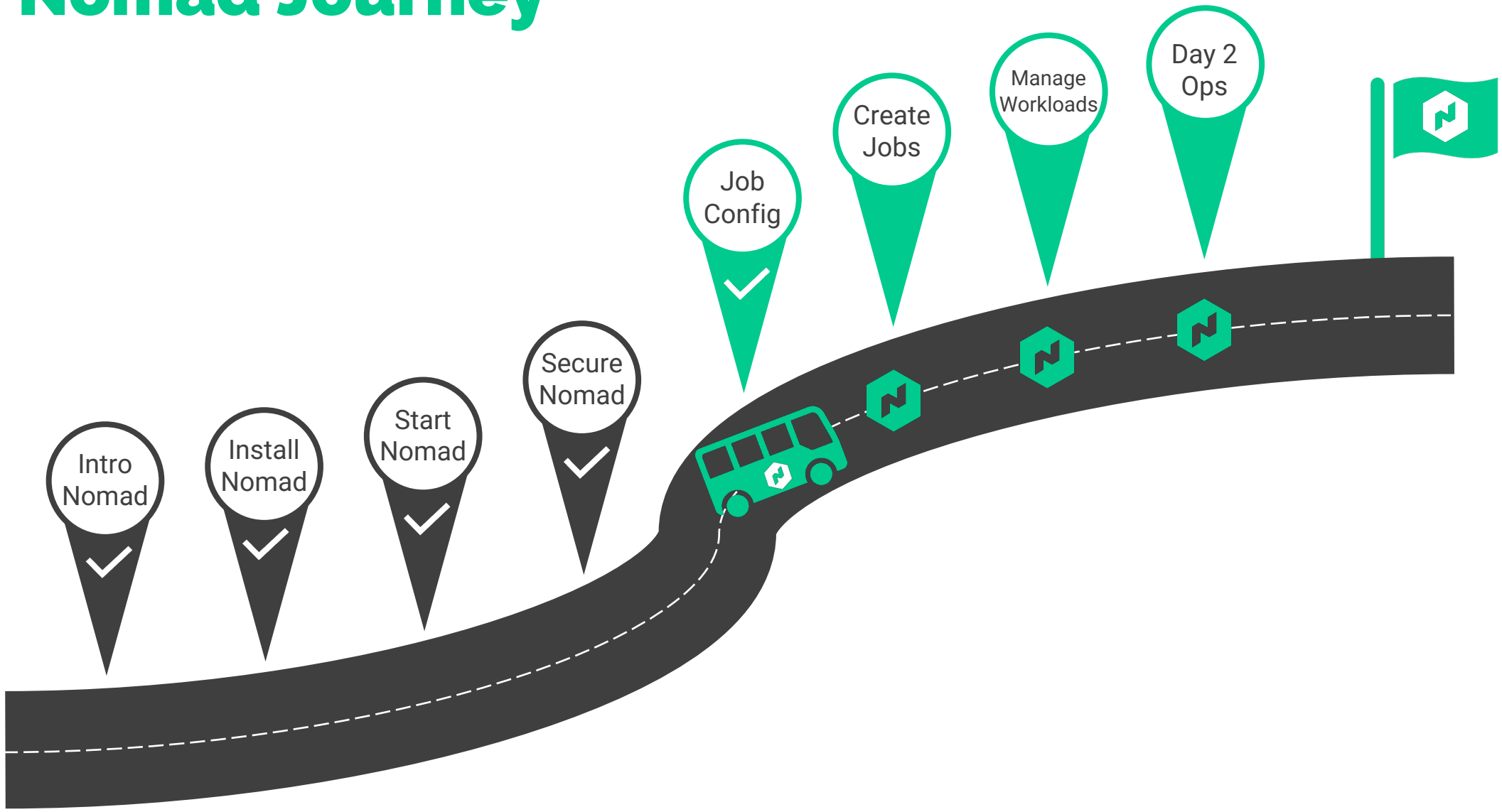
```
$ nomad validate tetris.nomad
Job validation successful
```

DEMO

Create a Job Specification

Running Our First Nomad Job

Nomad Journey



Nomad Job

- Once your job specification has been written, we can now create jobs and submit to Nomad to launch our application
- You can use the CLI or API to submit new jobs
- Use the command `nomad job run <file>` to submit the job to Nomad

TERMINAL

```
$ nomad job run [options] <file>
```

Nomad Job Plan

- Before you submit a "real" job, you can use the `nomad job run plan <file>` to perform a dry-run to determine what would happen if the job is submitted
- This is helpful to determine how the scheduler will react to the submission of this new job
- Can determine whether the job will run successfully, or it might show you that you have insufficient resources to run it

TERMINAL

```
$ nomad job run plan [options] <file>
```

Nomad Job Plan

TERMINAL

```
$ nomad job plan tetris.nomad
+/- Job: "tetris"
+/- Stop: "true" => "false"
    Task Group: "games" (1 create)
      Task: "tetris"

Scheduler dry-run:
- All tasks successfully allocated.

Job Modify Index: 10764
To submit the job with version verification run:

nomad job run -check-index 10764 tetris.nomad

When running the job with the check-index flag, the job will only be run if the
job modify index given matches the server-side version. If the index has
changed, another user has modified the job and the plan's results are
potentially invalid.
```

✓ PASSED

Nomad Job Plan

TERMINAL

```
$ nomad job plan tetris.nomad
+/- Job: "tetris"
+/- Stop: "true" => "false"
+/- Task Group: "games" (100 create)
  +/- Count: "1" => "100" (forces create)
      Task: "tetris"

Scheduler dry-run:
- WARNING: Failed to place all allocations.
  Task Group "games" (failed to place 94 allocations):
    * Resources exhausted on 3 nodes
    * Dimension "memory" exhausted on 3 nodes

Job Modify Index: 10764
To submit the job with version verification run:

nomad job run -check-index 10764 tetris.nomad

When running the job with the check-index flag, the job will only be run if the
job modify index given matches the server-side version. If the index has
changed, another user has modified the job and the plan's results are
potentially invalid.
```

✗ FAILED

Nomad Job

- Ok, let's submit our **tetris** job with just a count of "1" which we know will work fine

TERMINAL

```
$ nomad job run tetris.nomad
```

SAME COMMAND

TERMINAL

```
$ nomad run tetris.nomad
```

Nomad Job

TERMINAL

```
$ nomad job run tetris.nomad
==> 2023-01-04T15:10:12Z: Monitoring evaluation "ec4eb3c0"
    2023-01-04T15:10:12Z: Evaluation triggered by job "tetris"
    2023-01-04T15:10:13Z: Evaluation within deployment: "f5cbd676"
    2023-01-04T15:10:13Z: Allocation "83ff6abb" created: node "f55a64a7", group "games"
    2023-01-04T15:10:13Z: Evaluation status changed: "pending" -> "complete"
==> 2023-01-04T15:10:13Z: Evaluation "ec4eb3c0" finished with status "complete"
==> 2023-01-04T15:10:13Z: Monitoring deployment "f5cbd676"
  ✓ Deployment "f5cbd676" successful

    2023-01-04T15:10:31Z
    ID          = f5cbd676
    Job ID      = tetris
    Job Version = 0
    Status      = successful
    Description = Deployment completed successfully

    Deployed
    Task Group  Desired  Placed  Healthy  Unhealthy  Progress Deadline
    games       1        1       1        0          2023-01-04T15:20:29Z
```

Nomad Job

Use the command `nomad job status` to show the status of **all** jobs on the cluster

TERMINAL

```
$ nomad job status
ID       Type      Priority  Status    Submit Date
tetris   service   50        running   2023-01-04T15:10:12Z
vault    service   50        running   2022-12-27T15:09:14Z
```

I had another job already running on this cluster

Nomad Job

TERMINAL

```
$ nomad job status tetris
ID            = tetris
Name          = tetris
Submit Date   = 2023-01-04T15:10:12Z
Type          = service
Priority      = 50
Datacenters   = dc1
Namespace     = default
Status        = running
Periodic      = false
Parameterized = false

Summary
Task Group  Queued  Starting  Running  Failed  Complete  Lost  Unknown
games       0       0         1        0       0         0     0

Latest Deployment
ID          = f5cbd676
Status      = successful
Description = Deployment completed successfully

Deployed
Task Group  Desired  Placed  Healthy  Unhealthy  Progress Deadline
games       1        1       1        0          2023-01-04T15:20:29Z

Allocations
ID        Node ID   Task Group  Version  Desired  Status   Created    Modified
83ff6abb  f55a64a7  games       0        run      running  1h13m ago  1h13m ago
```

Use the command `nomad job status <job name>` to show the status of **all** jobs on the cluster

DEMO

Run Our First Nomad Job