What Cloud Engineers Could Learn from Clouds

Laura Santamaria, Lead Developer Advocate
Dell Technologies

DELL Technologies

@nimbinatus (hachyderm)

@nimbinatus (hachyderm)

Not these

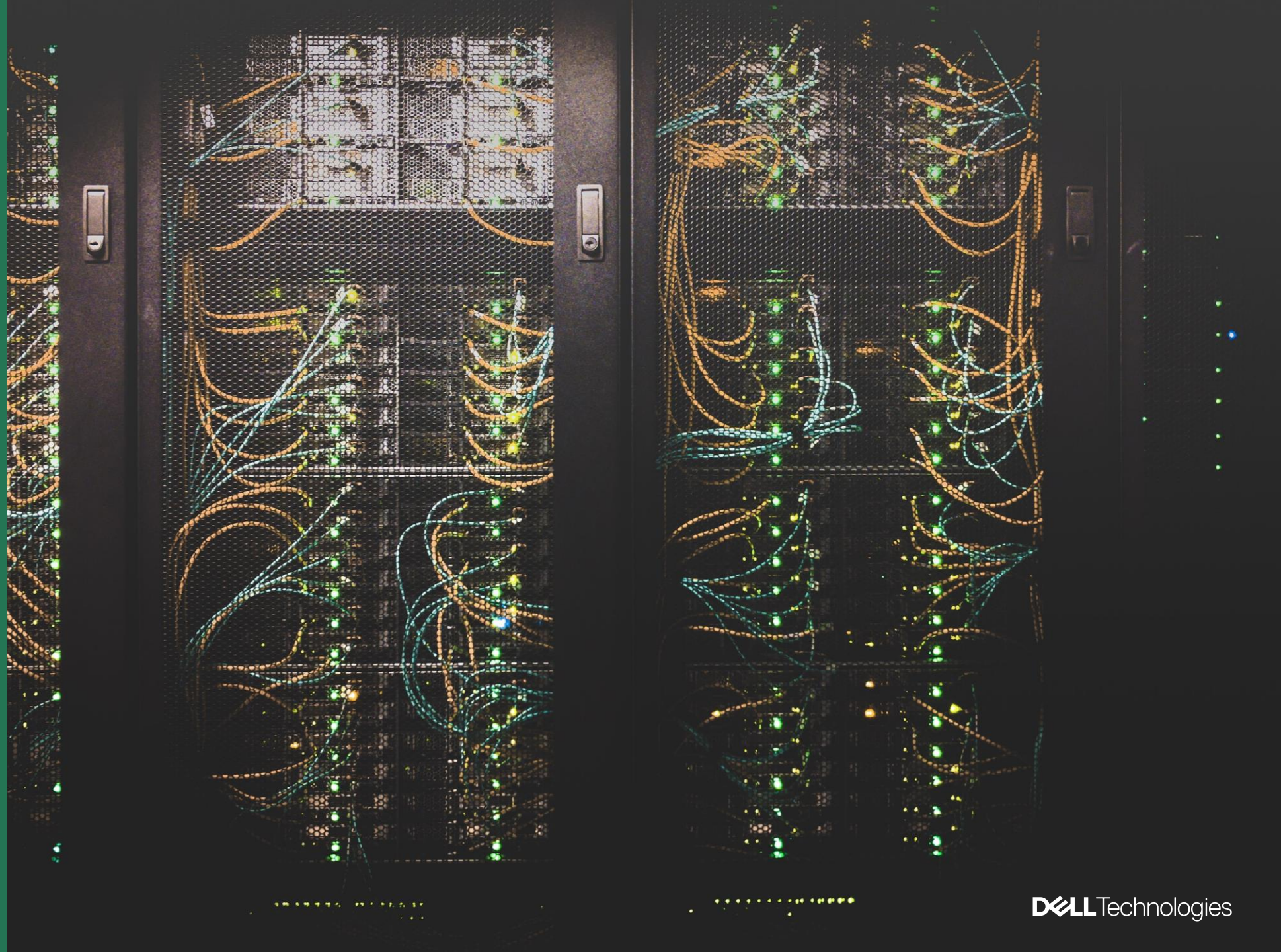


Photo by Taylor Vick on Unsplash

DELL Technologies

@nimbinatus (hachyderm)

These

Own work

DELL Technologies

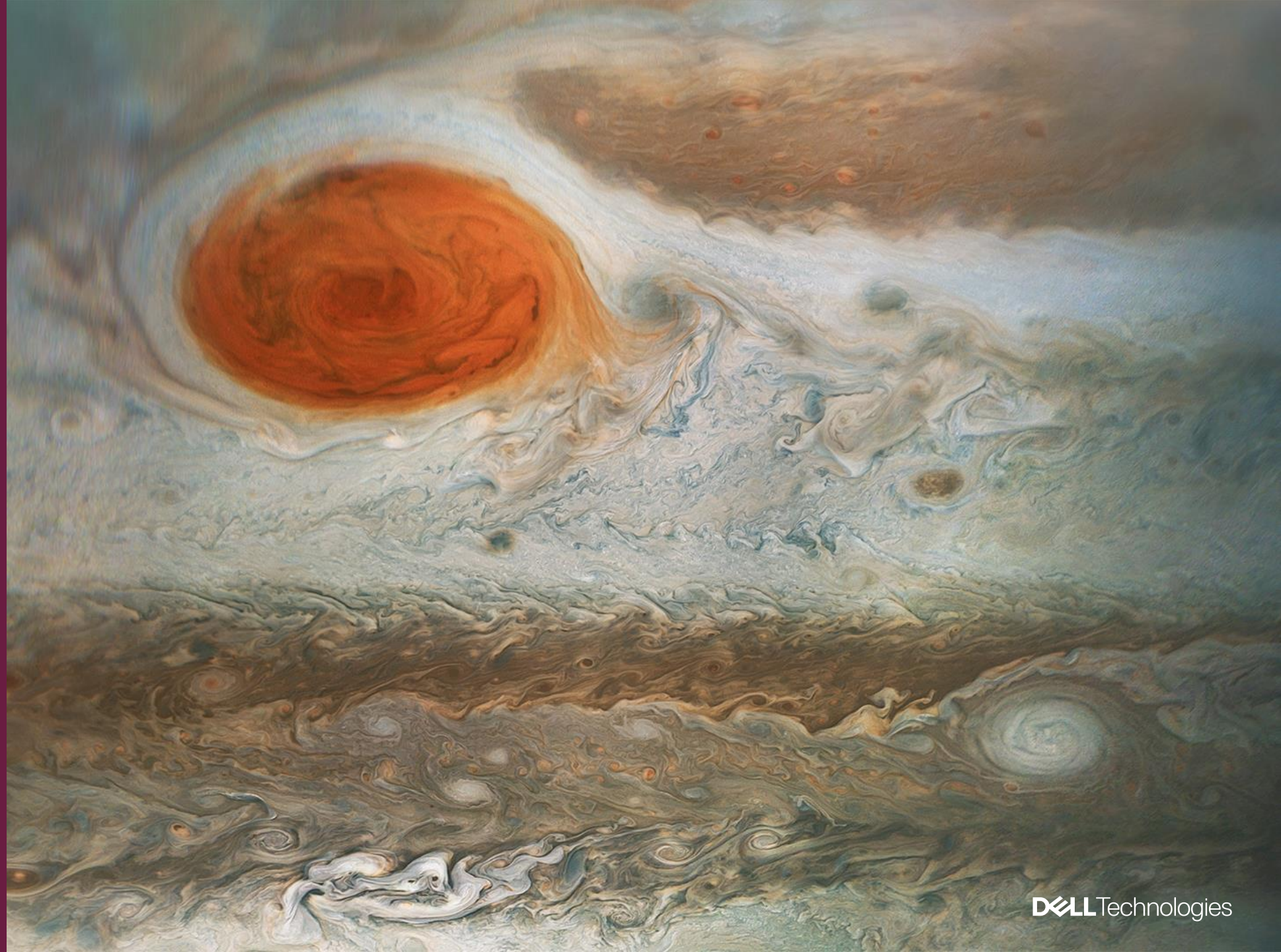
Intro

@nimbinatus (hachyderm)

History

@nimbinatus (hachyderm)

Planetary Atmospheres



Credits: NASA/JPL-Caltech/SwRI/MSSS/
Gerald Eichstädt /Seán Doran © CC NC SA

DELL Technologies

Clouds and Clouds

Definitions

@nimbinatus (hachyderm)

Complex Systems

@nimbinatus (hachyderm)

A cloud physics example

DELL Technologies

@nimbinatus (hachyderm)

A cloud computing example

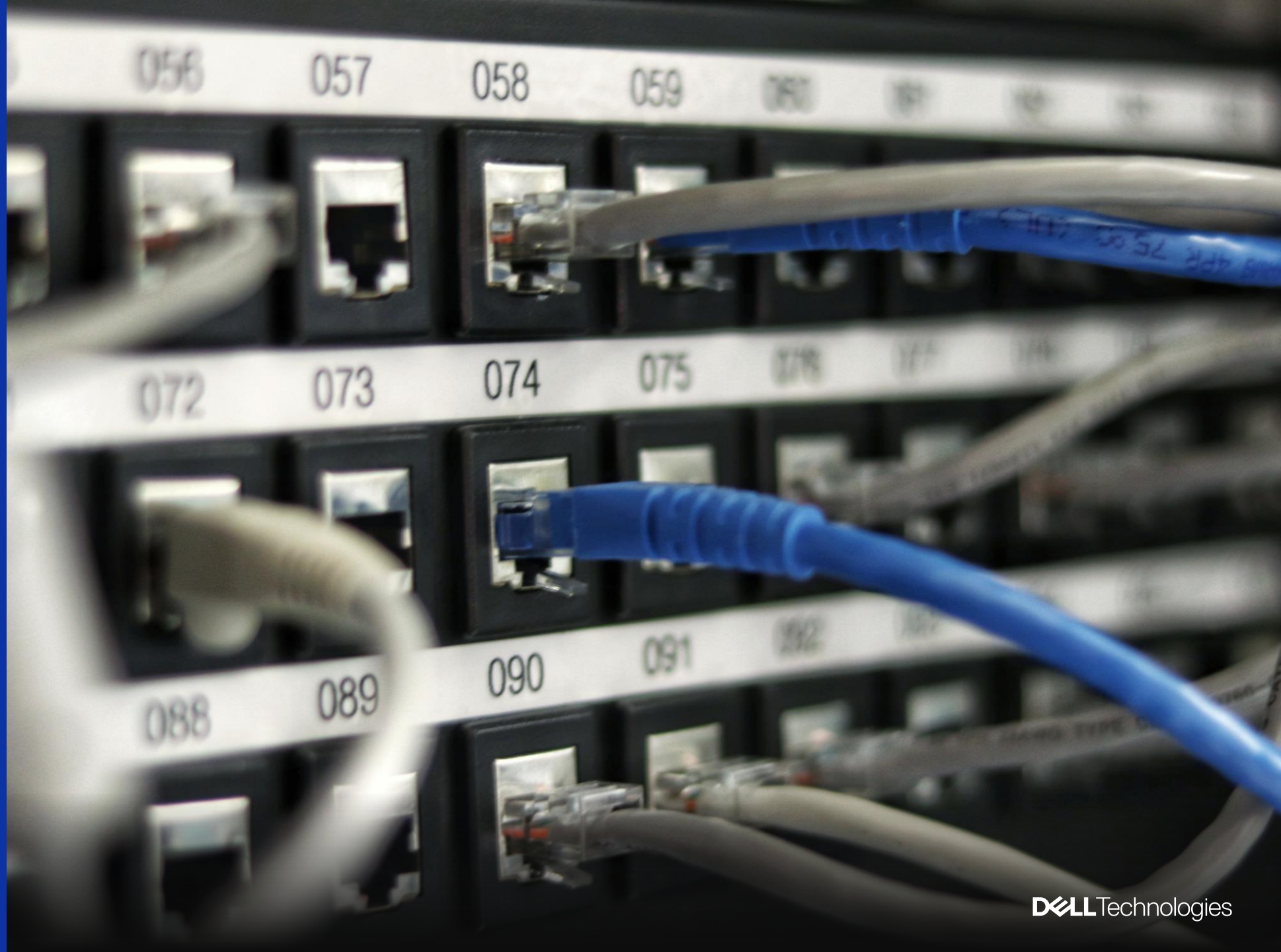


Photo by Jordan Harrison on Unsplash

DELL Technologies

Chaos and Chaotic Systems

@nimbinatus (hachyderm)

A cloud physics example

Photo by 2 Bro's Media on Unsplash

@nimbinatus (hachyderm)

A cloud computing example

Photo by Jon Tyson on Unsplash

DELL Technologies

Patterns

@nimbinatus (hachyderm)

A cloud physics example

Photo by Jordan Steranka on Unsplash

DELL Technologies

@nimbinatus (hachyderm)

A cloud computing example

Photo by Brian McGowan on Unsplash

DELL Technologies

All of this is what modeling is about.

Learning from clouds

@nimbinatus (hachyderm)

Understanding chaotic systems

@nimbinatus (hachyderm)

Observations

Photo by Daniel Lerman on Unsplash

@nimbinatus (hachyderm)

Experiments

Photo by CHUTTERSNAP on Unsplash

DELL Technologies

Observation and Experimentation

Build
Destroy
Repeat

Mindful Experimentation

Chaos Experiments

Modeling and predicting behavior

@nimbinatus (hachyderm)

@nimbinatus (hachyderm)

Data in, data out

Photo by NOAA on Unsplash

DELL Technologies

@nimbinatus (hachyderm)

Scale matters

Photo by NOAA on Unsplash

DELL Technologies

Modeling in computing systems

UML

TLA+

Alloy

TLA+ example (truncated)

From https://www.hillelwayne.com/modeling-deployments/

```
...
variables
  servers = {"s1", "s2", "s3"},
  load_balancer = servers,
  update_flag = [s \in servers |->
FALSE],
  updated = [s \in servers |->
FALSE];

define
  SameVersion ==
    \A x, y \in load_balancer:
      updated[x] = updated[y]

  ZeroDowntime ==
    \E server \in load_balancer:
      updated[server] /= UPDATING
end define

fair process update_server \in
servers
begin
  Update:
    await update_flag[self];
    updated[self] := UPDATING;
  FinishUpdate:
    updated[self] := TRUE;
end process;

fair process start_update =
"start_update"
  begin
   StartUpdate:
     load_balancer := {"s1"};
     update_flag := [s \in servers
|->
      IF s = "s1" THEN FALSE ELSE
TRUE];
   Transition:
     await \A s \in (servers \
load_balancer):
       updated[s] = TRUE;
     load_balancer := servers \
load_balancer;
     update_flag["s1"] := TRUE;
   Finish:
     await updated["s1"] = TRUE;
     load_balancer :=
load_balancer \union {"s1"};
end process;
...
```

TLA+ example (truncated)

From https://www.hillelwayne.com/modeling-deployments/

```
...
variables
    servers = {"s1", "s2", "s3"},
    load_balancer = servers,
    update_flag = [s \in servers |-> FALSE],
    updated = [s \in servers |-> FALSE];

define
  SameVersion ==
    \A x, y \in load_balancer:
      updated[x] = updated[y]

  ZeroDowntime ==
    \E server \in load_balancer:
      updated[server] /= UPDATING
end define

fair process update_server \in servers
begin
  Update:
    await update_flag[self];
    updated[self] := UPDATING;
  FinishUpdate:
    updated[self] := TRUE;
end process;

fair process start_update = "start_update"
  begin
   StartUpdate:
     load_balancer := {"s1"};
     update_flag := [s \in servers |->
      IF s = "s1" THEN FALSE ELSE TRUE];
   Transition:
     await \A s \in (servers \ load_balancer):
       updated[s] = TRUE;
     load_balancer := servers \ load_balancer;
     update_flag["s1"] := TRUE;
   Finish:
     await updated["s1"] = TRUE;
     load_balancer := load_balancer \union {"s1"};
end process;
```

TLA+ example (truncated)

From https://www.hillelwayne.com/modeling-deployments/

```
...
variables
  servers = {"s1", "s2", "s3"},
  load_balancer = servers,
  update_flag = [s \in servers |-> FALSE],
  updated = [s \in servers |-> FALSE];

define
  SameVersion ==
    \A x, y \in load_balancer:
      updated[x] = updated[y]

  ZeroDowntime ==
    \E server \in load_balancer:
      updated[server] /= UPDATING
end define

fair process update_server \in servers
begin
  Update:
    await update_flag[self];
    updated[self] := UPDATING;
  FinishUpdate:
    updated[self] := TRUE;
end process;

fair process start_update = "start_update"
  begin
   StartUpdate:
     load_balancer := {"s1"};
     update_flag := [s \in servers |->
      IF s = "s1" THEN FALSE ELSE TRUE];
   Transition:
     await \A s \in (servers \ load_balancer):
       updated[s] = TRUE;
```

TLA+ example (truncated)

From https://www.hillelwayne.com/modeling-deployments/

```
  servers = {"s1", "s2", "s3"},
  load_balancer = servers,
  update_flag = [s \in servers |-> FALSE],
  updated = [s \in servers |-> FALSE];

define
  SameVersion ==
    \A x, y \in load_balancer:
      updated[x] = updated[y]

  ZeroDowntime ==
    \E server \in load_balancer:
      updated[server] /= UPDATING
end define
```

```
fair process update_server \in servers
begin
  Update:
    await update_flag[self];
    updated[self] := UPDATING;
  FinishUpdate:
    updated[self] := TRUE;
end process;
```

```
fair process start_update = "start_update"
  begin
   StartUpdate:
     load_balancer := {"s1"};
     update_flag := [s \in servers |->
      IF s = "s1" THEN FALSE ELSE TRUE];
   Transition:
     await \A s \in (servers \ load_balancer):
       updated[s] = TRUE;
     load_balancer := servers \ load_balancer;
     update_flag["s1"] := TRUE;
   Finish:
     await updated["s1"] = TRUE;
     load_balancer := load_balancer \union {"s1"};
end process;
...
```

TLA+ example (truncated)

From https://www.hillelwayne.com/modeling-deployments/

```
...
variables
  servers = {"s1", "s2", "s3"},
  load_balancer = servers,
  update_flag = [s \in servers |-> FALSE],
  updated = [s \in servers |-> FALSE];

define
  SameVersion ==
    \A x, y \in load_balancer:
      updated[x] = updated[y]

  ZeroDowntime ==
    \E server \in load_balancer:
      updated[server] /= UPDATING
end define

fair process update_server \in servers
begin
  Update:
    await update_flag[self];
    updated[self] := UPDATING;
  FinishUpdate:
    updated[self] := TRUE;
end process;
```

```
fair process start_update = "start_update"
  begin
    StartUpdate:
      load_balancer := {"s1"};
      update_flag := [s \in servers |->
        IF s = "s1" THEN FALSE ELSE TRUE];
```

```
  Transition:
    await \A s \in (servers \ load_balancer):
      updated[s] = TRUE;
    load_balancer := servers \ load_balancer;
    update_flag["s1"] := TRUE;
  Finish:
    await updated["s1"] = TRUE;
    load_balancer := load_balancer \union {"s1"};
end process;
...
```

TLA+ example (truncated)

From https://www.hillelwayne.com/modeling-deployments/

```
variables
  servers = {"s1", "s2", "s3"},
  load_balancer = servers,
  update_flag = [s \in servers |-> FALSE],
  updated = [s \in servers |-> FALSE];

define
  SameVersion ==
    \A x, y \in load_balancer:
      updated[x] = updated[y]

  ZeroDowntime ==
    \E server \in load_balancer:
      updated[server] /= UPDATING
end define

fair process update_server \in servers
begin
  Update:
    await update_flag[self];
    updated[self] := UPDATING;
  FinishUpdate:
    updated[self] := TRUE;
end process;

fair process start_update = "start_update"
  begin
   StartUpdate:
     load_balancer := {"s1"};
     update_flag := [s \in servers |->
      IF s = "s1" THEN FALSE ELSE TRUE];
      Transition:
        await \A s \in
                  (servers \ load_balancer):
          updated[s] = TRUE;
        load_balancer := servers
                         \ load_balancer;
        update_flag["s1"] := TRUE;
   Finish:
     await updated["s1"] = TRUE;
     load_balancer := load_balancer \union {"s1"};
end process;
...
```

TLA+ example (truncated)

From https://www.hillelwayne.com/modeling-deployments/

```
...
variables
  servers = {"s1", "s2", "s3"},
  load_balancer = servers,
  update_flag = [s \in servers |-> FALSE],
  updated = [s \in servers |-> FALSE];

define
  SameVersion ==
    \A x, y \in load_balancer:
      updated[x] = updated[y]

  ZeroDowntime ==
    \E server \in load_balancer:
      updated[server] /= UPDATING
end define

fair process update_server \in servers
begin
  Update:
    await update_flag[self];
    updated[self] := UPDATING;
  FinishUpdate:
    updated[self] := TRUE;
end process;

fair process start_update = "start_update"
  begin
   StartUpdate:
     load_balancer := {"s1"};
     update_flag := [s \in servers |->
      IF s = "s1" THEN FALSE ELSE TRUE];
   Transition:
     await \A s \in (servers \ load_balancer):
       updated[s] = TRUE;
     load_balancer := servers \ load_balancer;
     update_flag["s1"] := TRUE;
     Finish:
       await updated["s1"] = TRUE;
       load_balancer := load_balancer
                        \union {"s1"};
end process;
...
```

Working with and around chaotic systems

@nimbinatus (hachyderm)

Models aren't perfect

Photo by Brett Jordan on Unsplash

@nimbinatus (hachyderm)

Model less complex things first

Photo by Marek Okon on Unsplash

@nimbinatus (hachyderm)

Models have limitations

Photo by Jiawei Zhao on Unsplash

Putting it all together

@nimbinatus (hachyderm)

DELL Technologies

Embrace failure

Trust your experience

Thanks

https://developer.dell.com

https://linktr.ee/nimbinatus

Feedback welcome!

→

@nimbinatus (hachyderm)