Notional PowerScale Architecture

I made this to build out a notional architecture for a PowerScale setup. Customer requires a solution which can do the following:

- Collect PCAP data at 25Gb/s
- Assuming 24/7 continuous storage, it must be able to store the PCAP for 30 days
- Dell should provide all networking, cabling, and sever equipment.
- Choice of PCAP collection software is left up to the vendor.
- The solution should provide a way to look up an arbitrary 5-tuple across the entire PCAP dataset and return a result within 5 seconds.

Decision Making Process for PowerScale

I began by identifying which chassis options would likely be most effective for the given scenario. To do this I started with the Isilon Gen 6 Cheat Sheet:

PowerScale - Isilon Gen 6 Cheat Sheet

Isilon Gen 6	Drive Type	Archive		Hybrid			Drive Type	Hybrid	Drive Type	All-Flash	
		A2000	A200	H400	H500	H5600		H600		F800	F810
Chassis Raw Capacity (4 Nodes)	2TB HDD	-	120 TB	120 TB	120 TB	-	600 GB SAS	72 TB	1.6 TB SSD	96 TB	-
	4TB HDD	-	240 TB	240 TB	240 TB	-			3.2 TB SSD	192 TB	-
	8TB HDD	-	480 TB	480 TB	480 TB	-			3.84 TB SSD	230 TB	230 TB
	10 TB HDD	800 TB	-	-	-	800 TB	1.2TB SAS	144 TB	7.68 TB SSD	460 TB	460 TB
	12TB HDD	960 TB	720 TB	720 TB	720 TB	960 TB			15.36 TB SSD	924 TB	924 TB
	16TB HDD	1280TB	960 TB	960 TB	960 TB	1280 TB		-		-	-
Drives Per Chassis		80	60	60	60	80		120		60	60
Cache (SSDs)		1 or 2 x 400 GB with 10 TB 1 or 2 x 800 GB with larger	1 or 2 x 400 GB with 2, 4 or 8 TB 1 or 2 x 800 GB with larger	1 or 2 800 GB, 1.6 TB or 3.2 TB	1 or 2 1.6 TB or 3.2 TB	1 or 2 x 3.2 TB with 10 TB 2 x 3.2 TB with 12TB		1 or 2 1.6 TB or 3.2 TB		-	-
ECC Memory Per Node		16** or 64 GB	16** or 64 GB	64 GB	128 GB	256 GB		256 GB		256 GB	256 GB
CPU		1 x Pentium D1508 2 Cores	1 x Pentium D1508 2 Cores	1 x Xeon D-1527 4 Cores	1 x E5-2630 v4 10 Cores	1 x E5-2680 v4 14 Cores		1 x E5-2680 v4 14 Cores		1 x E5-2697A v4 16 Cores	1 x E5-2697A v4 16 Cores
Front-End Networking Per Node		2 x 10/25 GbE	2 x 10/25 GbE	2 x 10/25 GbE	2 x 10/25/40 GbE	2 x 10/25 GbE		2 x 10/25/40 GbE		2 x 10/25/40 GbE	2 x 10/25/40 GbE
Back-End Networking Per Node		IB or 2 x 10GbE	IB or 2 x 10GbE	IB or 2 x 10GbE	IB or 2 x 40GbE	IB or 2 x 40GbE		IB or 2 x 40GbE		IB or 2 x 40GbE	IB or 2 x 40GbE
Dimensions		H: 17.8 cm (7") W: 44.8 cm (17.6") D: 107.1 cm (42.2")		H: 17.8 cm (7") W: 44.8 cm (17.6") D: 95.5 cm (37.6")		H: 17.8 cm (7") W: 44.8 cm (17.6") D: 107.1 cm (42.2")		H: 17.8 cm (7") W: 44.8 cm (17.6") D: 95.5 cm (37.6")		H: 17.8 cm (7") W: 44.8 cm (17.6") D: 95.5 cm (37.6")	

Figure 1: PowerScale Cheat Sheet

In our notional architecture we need $25 * 60^2 * 24 * 31/8 = 8,370,000$GBs of storage or 8.37PB. Our major bottleneck is going to be storage space. For our solution we will use Google Stenographer to capture the traffic.

The first thing we have to consider is how much space we'll actually need to effectively store 8.37PB. In the interest of being a bit more investigative, I decided to research how Reed-Solomon, the algorithm by which we store data, actually works.

Calculating Total Space Required

Isilon Sizing: Sizing for Accurate Capacity by Bhupendra Singh

Understanding Reed-Solomon Error Correction CMU Reed-Solomon Codes

I took this article and adapted the sections I found confusing to clarify the explanation.

Reed-Solomon is represented with the notation `RS(n,k)`. This includes `s-bit` symbols which I'll explain in a moment. Collectively this means that the Reed-Solomon (RS) encoder takes k data symbols (whatever the user wants to store), each of length s and adds some number of parity symbols which collectively create an n symbol codeword. There are $n - k$ parity symbols, each with length s which allow us to correct for up to t symbols that contain errors in a codeword where $2t = n - k$.
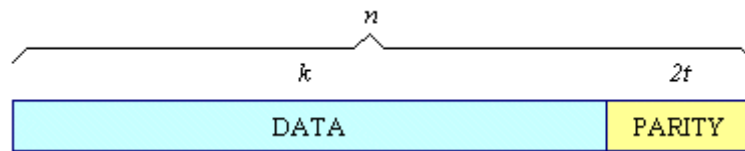

Figure 2: https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_code_word.gif

Image from CMU Reed-Solomon Codes

Example

A popular Reed-Solomon code is RS(255,223) with 8-bit (one byte) symbols. Said another way, it is 255 1 byte symbols (the total size of the codeword) and 223 1 byte symbols of that are data symbols (the actual payload). Of the total, 223 bytes are data and 32 bytes are parity. This breaks down into: $n = 255, k = 223, s = 8, 2t = 32, t = 16$ This means that the decoder can correct up to 16 bytes (16 symbols) of errors anywhere in our 223 bytes of data without issue.

Symbol Error: A symbol error occurs if any number of bits in a single symbol (in our example, a data byte) are wrong. It doesn't matter how many bits are wrong; it could be one or all of them.

Decoding Reed-Solomon algebraic decoding procedures can correct errors and erasures. An erasure occurs when the position of an erred symbol is known. A decoder can correct up to t errors or up to 2t erasures.

What is an Erasure? In coding theory, an erasure code is a forward error correction (FEC) code under the assumption of bit erasures (rather than bit errors), which transforms a message of k (remember that's our data) symbols into a longer message (code word) with n symbols such that the original message can be recovered from a subset of the n symbols. The fraction $r = \frac{k}{n}$ is called the *code rate*. The fraction $\frac{k'}{k}$, where k' denotes the number of symbols required for recovery, is called *reception efficiency*.

The part that wasn't clear to me The particular scheme in use is unknown

- erasure coding is a general concept. Like an RFC, there are mathematical principals to which it must adhere, in this case it must obey the rules set out by RS, but there isn't a specific defined method.

When a codeword is decoded, there are three possible outcomes:

1. If $2s + r < 2t$ (s errors, r erasures) then the original transmitted code word will always be recovered
2. If it isn't recovered, the decoder will detect that it cannot recover the original code word and indicate this fact OR
3. The decoder will mis-decode and recover an incorrect code word without any indication.

The probability of each of the three possibilities depends on the particular Reed-Solomon code and on the number and distribution of errors.

My Simple Mathematic Intuition of a Reed Solomon Encoder/Decoder
Based on this lecture. They used colorized diagrams with the matrix math implied by a multiplication of the colors whereas I reinterpreted that as a formal matrix equation.

Using linear algebra, a simple Reed Solomon code follows the canonical pattern $Ax = b$. Assume you have n data words $d_1, ..., d_n$ and m coding words $c_1, ..., c_m$. Encoding and decoding will use an $n+m$ by n coding matrix B.

Let B some matrix such that $B < d_1, \ldots, d_n > = < d_1, \ldots, d_n, c_1, \ldots, c_m >$ where the first n rows of B will form the identity matrix. It doesn't have to actually be the identity matrix, the matrix B simply must adhere to the property that deleteing any m rows result in an invertible matrix for the math to work out in the next part. It's worth saying that when we delete m rows the resulting matrix will be square which is a necessary condition to being invertible along with each of the columns being linearly independent (no one column is an integer multiple of another column). My notes on invertibility and linear independence

$$\begin{bmatrix} \ldots I_{nxn} \ldots \\ \ldots RAND_{mxn} \ldots \end{bmatrix}$$

The resulting matrix is an $n+m$ by n with the top part being some nxn matrix and the bottom part being an mxn matrix for the code word portion.

To decode, you would receive n words. By some method of your choosing (it doesn't actually matter what) you know what rows from B correspond to those n words. See page 14 of the lecture for a better graphic since I'm not doing the explanation much justice here for the sake of brevity.

Then you form the equation $B'^{-1}Bx = B'^{-1}b$ where B' is an invertible matrix created from the rows of B which correspond to the correct rows of n and x is the received data. The result is the original data. Various systems exist to make it so that the original matrix has properties which make it so that you would be able to reconstruct the original data if one piece were missing. A very naive way

could be the sum of the elements of the original codewords in n. If you know in what word the failure occurred (an erasure) you could use the total value to recreate the missing element.

Return to Sizing PowerScale

Thanks to the above, we now understand what N and M are in the context of OneFS. Here they just directly correlate to drives. I'll assume that we set N and M to maximum for a stripe width of 20. PowerScale has a block size of 8KB so our minimum file size is 128KB. Less than that and our files are mirrored rather than employing erasure coding which we want to avoid. In our use case, we can control the size of the PCAP files written so this isn't a problem. Subsequently, we should be operating close to the theoretical max efficiency.

Using the sizing tool it comes up with a number of viable configurations. Based on a guess, I suspect the following configuration to be roughly the most cost effective.

It is worth noting for capacity planning that I assumed 100% utilization which is impossible. That said, in the given use case, in the many times I've heard it, never once has 100% utilization actually been reached. Subsequently, in the interest of being cost effective I am cutting it closer than would truly be possible. I also only protected it at 1n because customers in this particular scenario rarely want additional protection. The risk of losing PCAP is worth it when offset by the cost of the additional protection. Some even want no redundancy which isn't possible in the PowerScale. Between the cheat sheet and the sizing tool I settled on a solution employing 32 H5600s.

PowerScale Primary Cluster

Description	OneFS Version
Isilon Scale-Out NAS	**9.2.1**

Node Pools	Backend Networking Type
1	**Ethernet**

Effective Capacity	Raw Capacity
8654.03 TiB	**9313.23 TiB**
Usable Capacity at 100%	Usable Capacity at 85%
8654.03 TiB	**7355.93 TiB**

CAPACITY & DISKS ⌃

NodePool #1

32 x H5600	20 x 16 TB	HDD

POWER DETAILS ⌃

Typical power consumption @ 240v: 13344.00 watts

Max power consumption @ 240v: 15584.00 watts

Typical thermal rating: 45024.00 BTU

Max thermal rating: 53174.72 BTU

ADDITIONAL DETAILS ⌃

Aggregate rack units: 32U

Cluster weight: 2280.00 lb

SizerID	pscale_17956

32 x H5600

20 x 16 TB HDD | 2 x 3.2 TB Cache SSD | RAM 256 GiB

CAPACITY ^

Effective Capacity at 100%	Raw Capacity
8654.03 TiB	9313.23 TiB
Usable Capacity at 100%	Usable Capacity at 85%
8654.03 TiB	7355.93 TiB

Data Reduction Ratio	Protection	Storage Efficiency
1:1	+1n	93.75%

NETWORKING ^

NDMP NIC: 2X10GbE + 2X8G FC W/ OPTICS
2x40GbE (QSFP+) Back End W/O OPTICS

POWER DETAILS ^

Typical power consumption @ 240v: 13344.00 watts
Max power consumption @ 240v: 15584.00 watts
Typical thermal rating: 45024.00 BTU
Max thermal rating: 53174.72 BTU

ADDITIONAL DETAILS ^

Aggregate rack units: 32U
NodePool weight: 2280.00 lb

Networking

Based on our documentation, each node should be cabled in the following manner:

That means 2x connections for both the backend and frontend networks which means for a 32 node setup 64 total connections each on the backend and frontend networks. Following is the breakdown of the network equipment we would need:

- 4 Z9264s to cover both backend and frontend networks assuming the customer already has spine switches available. If not we would need to

- Port 1 on the Isilon to port 1 on the primary external switch (ext1)
- Port 2 on the Isilon to port 1 on the secondary external switch (ext2)
- Port 3 on the Isilon to Port 1 on the primary internal switch (int1)
- Port 4 on the Isilon to port 1 on the secondary internal switch (int2)

Add additional Isilon nodes using open ports in ascending order (left to right) on the switch, as illustrated in the following figure.

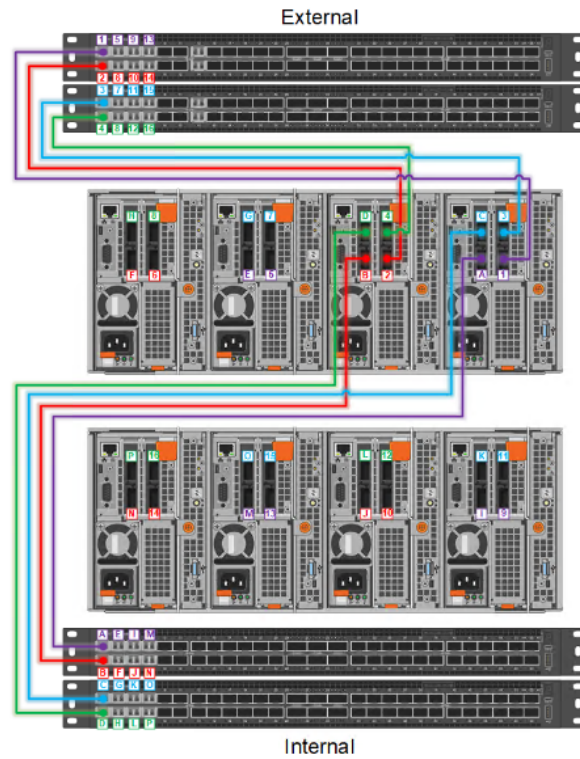Figure 12. Isilon node cabling diagram

Figure 3: Wiring Diagram

7

provide those as well.

- If we needed to include spines we would also need the 100GbE uplink cables
- 130 MPO cables - 4 per node plus 2 for an R740 I'd put on the frontend to actually ingest the traffic.
- For this use case we could use either 10Gb/s or 25Gb/s receivers on the frontend - it depends on how the customer plans to ingest the traffic. We'll say that any node should be able to support a full PCAP load and act as the "captain" (as the technical documentation calls it) for full line rate. Subsequently we'd need 66 25Gb/s ethernet transceivers.
  - Depending on existing infrastructure, we would need an additional 33 ethernet transceivers for all of the management connections plus an additional 33 ethernet cables.

For the topology, it would largely depend on what the customer is doing. In the past I've typically ran L2 leaf-spine with VLT because most customers these days don't have much routing going on (or at least not the ones I've been sent to). That said, we could just as easily put in a L3 leaf-spine config with OSPF.
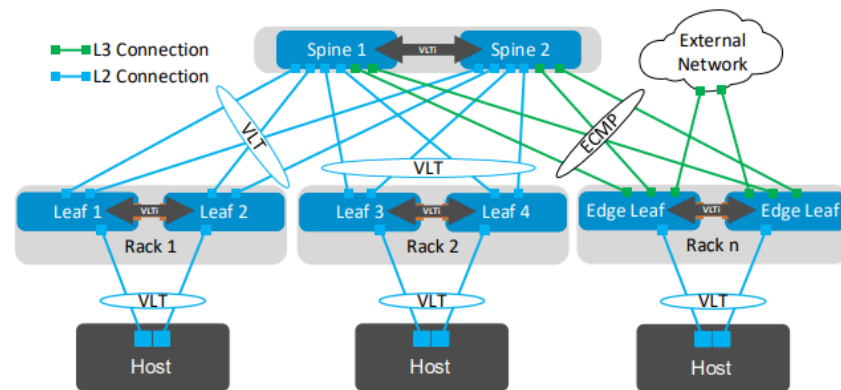

Figure 4: Spine-Leaf Architecture

We would have at least three VLANs - one for the backend, one for the frontend, and one for management.

Software Functionality

In our notional use case our software is Google Stenographer but realistically it could be any capture program. The import piece to consider from a storage perspective is that this setup could potentially ingest up to 25Gb/s which the PowerScale would see all in one file stream. This means our "captain" box would actually see a single 25Gb/s write coming in. Our notional architecture will

support this with the 25Gb/s hitting the frontend via whatever mount we choose (probably NFS) and then it would distribute it out to the supporting nodes as described in the technical guide.
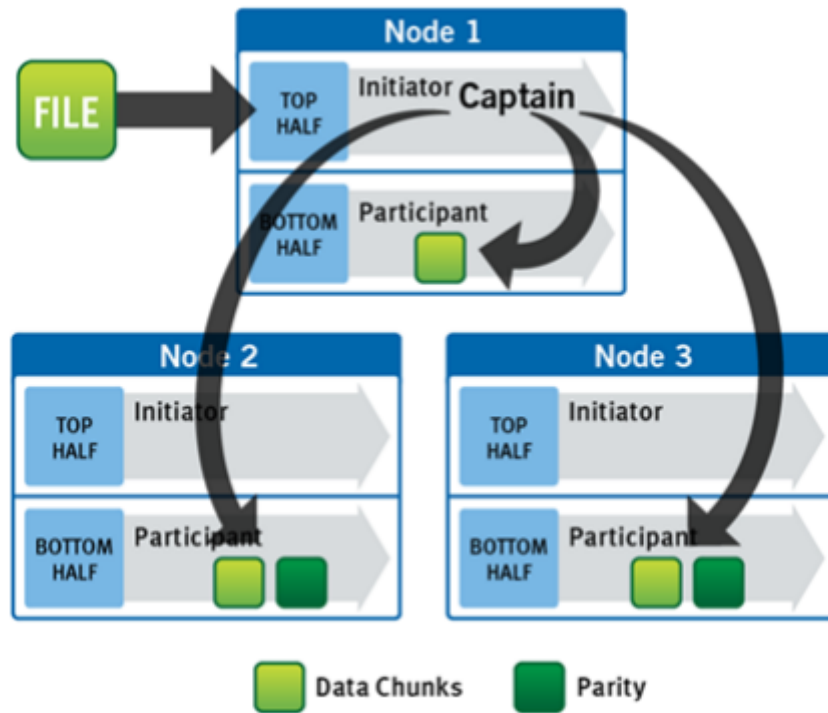

Figure 7: A File Write Operation on a 3-node Cluster

Figure 5: File Write Operations

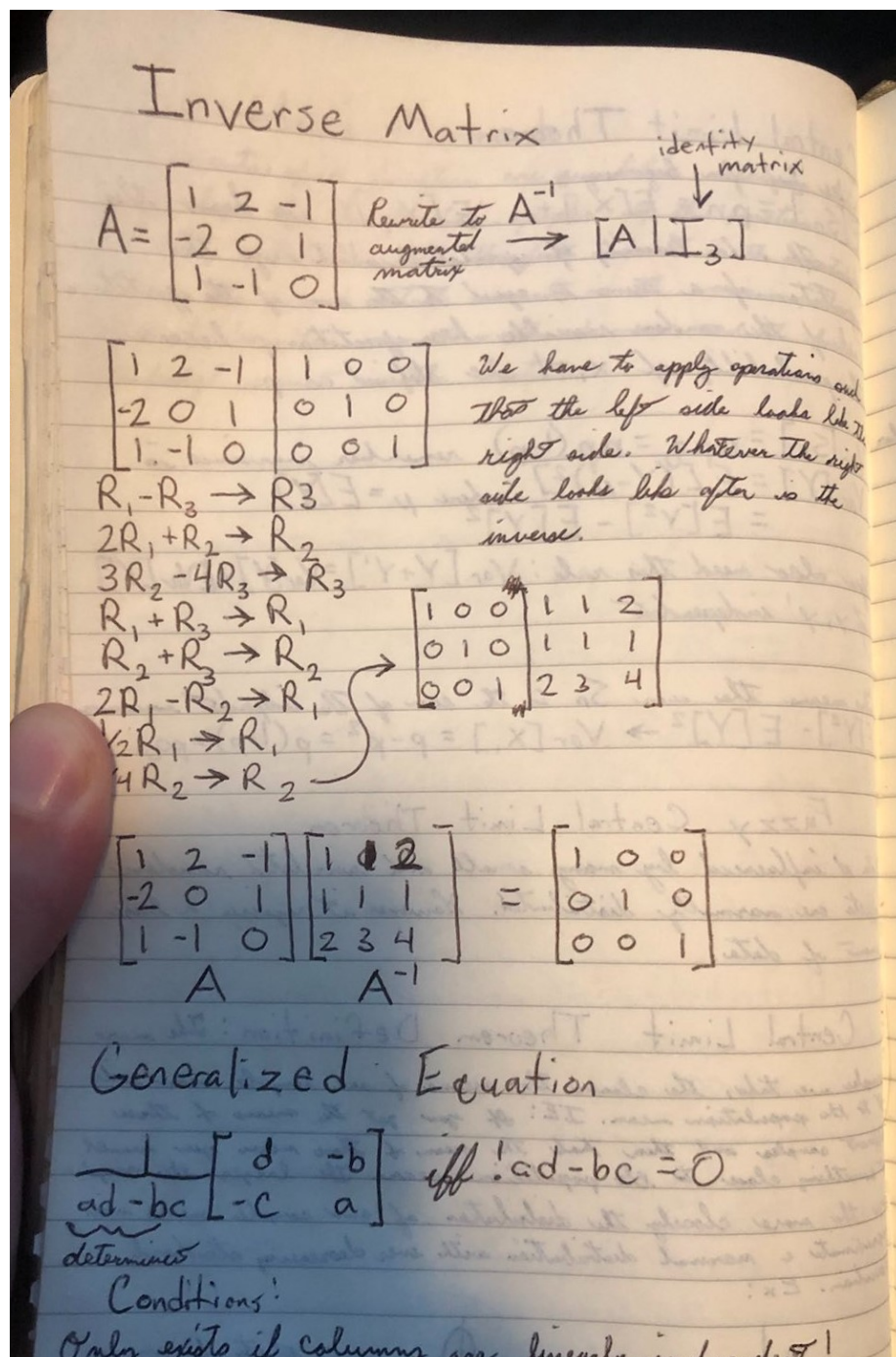## Invertibility and Linear Independence

Inverse Matrix

$$A = \begin{bmatrix} 1 & 2 & -1 \\ -2 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}$$

Rewrite to augmented matrix $\longrightarrow$

identity matrix $\downarrow$

$[A \mid I_3]$

$$\begin{bmatrix} 1 & 2 & -1 & | & 1 & 0 & 0 \\ -2 & 0 & 1 & | & 0 & 1 & 0 \\ 1 & -1 & 0 & | & 0 & 0 & 1 \end{bmatrix}$$

We have to apply operations onto so that the left side looks like I_3 the right side. Whatever the right side looks like after is the inverse.

$R_1 - R_3 \rightarrow R_3$
$2R_1 + R_2 \rightarrow R_2$
$3R_2 - 4R_3 \rightarrow R_3$
$R_1 + R_3 \rightarrow R_1$
$R_2 + R_3 \rightarrow R_2$
$2R_1 - R_2 \rightarrow R_1$
$\frac{1}{2}R_1 \rightarrow R_1$
$4R_2 \rightarrow R_2$

$$\longrightarrow \begin{bmatrix} 1 & 0 & 0 & | & 1 & 1 & 2 \\ 0 & 1 & 0 & | & 1 & 1 & 1 \\ 0 & 0 & 1 & | & 2 & 3 & 4 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 1 & 2 & -1 \\ -2 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} 1 & 1 & 2 \\ 1 & 1 & 1 \\ 2 & 3 & 4 \end{bmatrix}}_{A^{-1}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Generalized Equation

$$\underbrace{\frac{1}{ad-bc}}_{determinant} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \quad iff \; ! \, ad - bc = 0$$

Conditions:
Only exists if columns are linearly independent!

Linearly Independent

Anytime there is no constant such that one vector is equal to another, those vectors are said to be linearly independent. For example

$\begin{bmatrix} 7 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ -1 \end{bmatrix}$ which would be perpendicular lines

Formal definition of linear dependence
Linearly dependent iff $c_1 v_1 + c_2 v_2 + \ldots + c_n v_n$
$= \vec{0} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$ for some c_i's not all zero

Linear Subspace

Subspace of R^n
↳ subspace

V ← subset of R^n and V contains $\vec{0}$
$\vec{X}$ in V must satisfy $c\vec{X}$ in V - this is called closure under scalar multiplication

if $\vec{a}$ in V and $\vec{b}$ in V then $\vec{a} + \vec{b}$ in V
this is called closure under addition.

All of these things imply a subspace