Graphviz Graphs from text

START HERE

`digraph G { a → b }`
Directed graph

`graph G { a -- b }`
Undirected graph

`dot -Tpng in.gv -o out.png`
Render to PNG

`a [label="Start"]`
Label a node

`rankdir=LR`
Lay out left to right

`a → b [label="yes"]`
Labelled edge

`{rank=same; a; b}`
Force same row

`subgraph cluster_x { }`
Draw a box group

DOT text .gv / .dot → `dot` → **Layout** positions → `-T<fmt>` → **Output** svg, png, pdf

CONCEPTS

Edges use → in digraph and -- in graph; mixing them is a syntax error.

Pick the engine by graph shape: dot for hierarchies, neato/fdp for networks, circo for rings.

Quote any ID with spaces or dashes: "my node". Keywords like node or edge need quotes too.

COMMAND LINE

`dot -T<fmt> <file> -o <out>`
Render with an explicit format

`dot -Tsvg <file> > <out>.svg`
Render SVG to stdout redirect

`dot -Kneato -Tpng <file> -o <out>`
Choose layout engine by name

`neato / fdp / circo / twopi`
Alternative layout programs

`dot -Gkey=value <file>`
Set a graph attribute

COMMAND LINE (cont.)

`dot -V`
Show version

`dot -T:`
List available output formats

GRAPH BASICS

`digraph G { ... }`
Directed graph body

`strict digraph G { ... }`
Merge duplicate edges

`a → b → c`
Chain of edges

`a → {b c d}`
One source, many targets

`a [label="Name"]`
Node with display text

`a → b [label="x"]`
Edge with a label

`// ...  /* ... */  # ...`
Comment forms
 # is only at line start in some contexts; // is the safe choice.

`"two words"`
Quote IDs with spaces

`label="line1\nline2"`
Line break in a label
\l and \r left- and right-justify the line.

GRAPH BASICS (cont.)

`node [shape=box]`
Set node defaults from here on

`graph [bgcolor=white]`
Set graph-wide attribute

NODES

`shape=box`
Rectangle node

`shape=circle / diamond`
Circle or decision diamond

`shape=record`
Field-split box
Fields: label="{a|b|<p>c}"; | splits, { } flips direction.

`shape=plaintext / none`
Text with no outline

`style=filled fillcolor=<c>`
Fill node colour

`color=<c> penwidth=<n>`
Outline colour and width

`fontname=<f> fontsize=<n>`
Font face and size

`URL="<url>"`
Clickable node in SVG

EDGES

`style=dashed / dotted`
Line style

EDGES (cont.)

`dir=both`
Arrowheads on both ends

`dir=none`
No arrowheads

`arrowhead=<type>`
Arrow style at head
Examples: normal, vee, dot, odot, none, diamond.

`constraint=false`
Edge ignored for ranking

`weight=<n>`
Keep edge short and straight

`splines=ortho`
Right-angle edge routing

`a:p1 → b:p2`
Connect record ports

LAYOUT AND GROUPS

`rankdir=LR`
Left to right (TB default)

`{rank=same; a; b}`
Put nodes on one rank

`subgraph cluster_<n> { ... }`
Box group; name must start cluster

LAYOUT AND GROUPS (cont.)

`label="<t>"; style=filled;`
Style the cluster box

`nodesep=<in> ranksep=<in>`
Spacing between nodes and ranks

`ratio=<r> size="<w>,<h>"`
Constrain overall size

`overlap=false`
Spread nodes in neato/fdp

`layout=neato`
Choose engine inside the file

`newrank=true`
Rank across clusters correctly

Layout engines

Engine	Best for
`dot`	Layered hierarchies, flows
`neato`	Spring model, small graphs
`fdp`	Springs with clusters
`sfdp`	Very large graphs
`circo`	Circular layouts
`twopi`	Radial from a root

COLOURS AND LABELS

`color="#rrggbb"`
Hex colour

`label=<<b>bold</b>>`
HTML-like label

COLOURS AND LABELS (cont.)

`label=<<table>...</table>>`
Table in a node

RECIPES

Flowchart left to right

```
digraph G {
  rankdir=LR; node [shape=box,
style=rounded];
  start → build → test →
deploy;
  test → build [label="fail",
style=dashed];
}
```

Cluster with shared rank

```
digraph G {
  subgraph cluster_a
{ label="Backend"; api; db; }
  web → api → db;
  {rank=same; web; cache}
}
```

Record node with ports

```
digraph G {
  node [shape=record];
  t [label="{name|<id>id|
<fk>owner}"];
  u [label="{user|<id>id}"];
  t:fk → u:id;
}
```

Batch render a folder

```
for f in *.gv; do
  dot -Tsvg "$f" -o
"${f%.gv}.svg"
done
```