AN ILLUSTRATED GUIDE

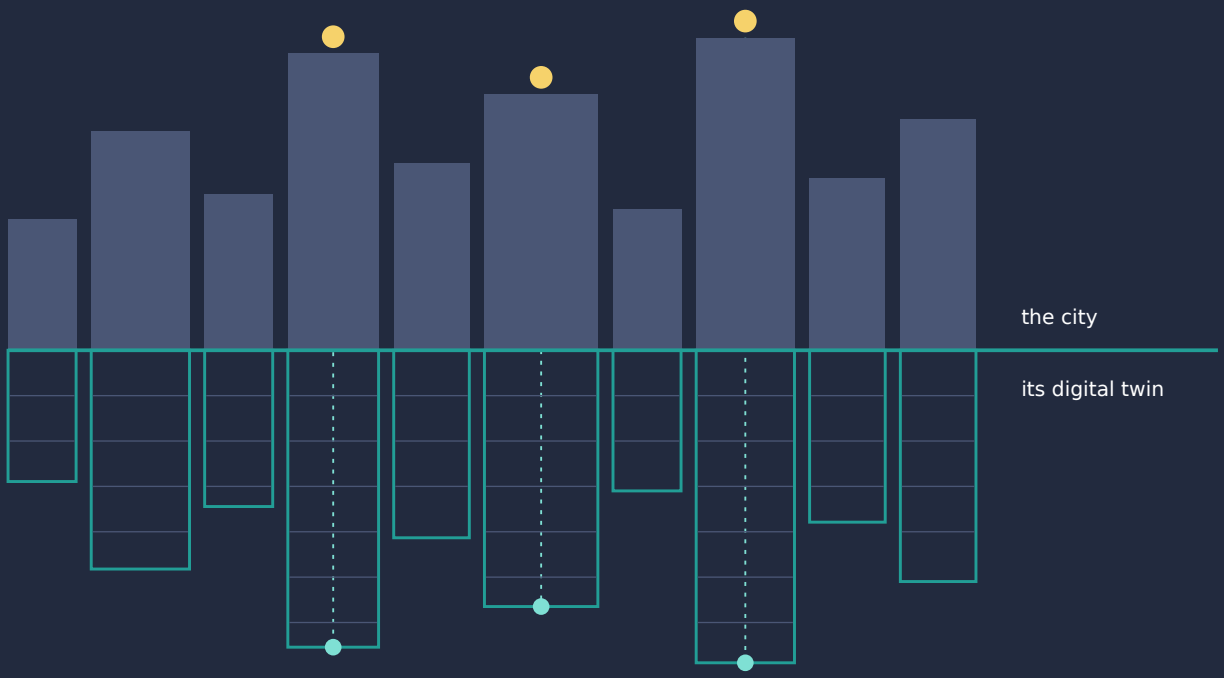
Urban Digital Twins

A friendly guide to living digital copies of cities and how we use them

Plain English, pictures first, a mini twin of one street built in code, and a glossary, a quiz and an eight-week plan.

Sixteen short chapters · Quiz · Glossary · Eight-week plan

Prompted and directed by Mahmoud A.
Written and illustrated by Claude (Anthropic)



How to Use This Guide

Read it in order, or jump to what you need.

This guide explains **urban digital twins**. An urban digital twin is a digital copy of real streets, buildings and networks that stays connected to the real place. Data flows in, so the copy changes when the city changes. People use the copy to ask questions, test ideas and run services with fewer surprises.



Figure 1. The whole idea in one picture. Data flows from the city into its twin, and decisions flow back out.

Who it is for

- Anyone curious about how cities use 3D models, sensors and simulation.
- Planners, designers, engineers and data people who want a clear picture.
- Self-learners. You need no maths beyond averages, and no coding. Start with Chapters 1 and 2.

How each chapter works

- A navy banner opens each chapter. A plain explanation comes first, then a picture, then an example.
- **TIP** boxes give practical advice. **SAY IT LIKE THIS** boxes give a sentence you could use out loud. **WATCH OUT** boxes flag common mistakes. **IN ONE LINE** boxes sum up each chapter.
- New words are in bold the first time they appear. The glossary at the back collects them.

Where the numbers come from

Every number in the worked examples comes from a small Python script that builds this guide. The sensors, buildings, traffic and storm are made up for teaching and do not describe any real place. When a chapter shows a result, it says that the script computed it. The guide quotes no statistics about real projects.

Contents

Sixteen short chapters and a toolbox at the back.

1 **What Is an Urban Digital Twin?**
A living copy of a city that stays connected to the real place.

2 **A City in Six Layers**
The parts every twin is built from.

3 **The 3D City Model**
The body of the twin, with its shape and its meaning.

4 **Sensors and the Internet of Things**
The senses of the twin, and why readings need checking.

5 **The Data Platform**
The plumbing that joins everything up.

6 **Simulation and Analytics**
Explaining the past and predicting what comes next.

7 **Seeing, Deciding, Acting**
Visualisation, decisions and the feedback loop.

8 **How Fresh Is the Data?**
Update rates, and why they matter.

9 **Maturity Levels**
From a static map to a twin that helps run the city.

10 **Standards in Plain Words**
The shared languages that let the parts work together.

11 **What Twins Are Used For**
Mobility, energy, heat, flooding and planning.

12 **Designing with What-If Scenarios**
Testing ideas in the copy before building them in the city.

13 **A Mini Twin of One Street**
A tiny twin built in code, step by step.

14 **Governance, Privacy and Cost**
The questions that decide whether a twin lasts.

15 **Common Pitfalls**
Mistakes that are easy to make and easy to avoid.

16 **Try It Yourself**
Free tools, a short recipe and an eight-week plan.

At the back: Myths and Facts · Quick Quiz · Glossary · Cheat Sheet · Further Reading

1 What Is an Urban Digital Twin?

A living copy of a city that stays connected to the real place.

An **urban digital twin** is a digital copy of a city, or of part of one, that stays linked to the real place. It holds the shape of buildings, streets, trees and pipes. It also receives fresh data, such as traffic counts, temperatures and energy use. Because of that link, the copy can show what is happening now, help explain why, and test what might happen next.

The idea began in manufacturing and aerospace. Engineers kept a digital copy of a machine, such as an engine, linked to the real one. Sensors on the machine updated the copy, and the copy warned of wear before parts failed. Cities borrowed the idea. A city is far bigger and messier than an engine, yet the logic is the same.

Model, shadow, twin

People use the words “digital twin” loosely. A helpful way to sort things out is to ask how data moves between the real thing and its copy. Researchers in manufacturing proposed three names for this, and they work well for cities too.

- **Digital model.** A digital copy with no automatic link. Someone updates it by hand, now and then. A 3D city model made from one aerial survey is a digital model.
- **Digital shadow.** Data flows automatically from the city into the copy, but nothing flows back on its own. A live traffic map is a digital shadow. It follows the city the way a shadow follows you.
- **Digital twin.** Data flows automatically in both directions. The copy learns from the city, and its results feed back into decisions or controls. A system that reads traffic sensors, predicts queues and retimes traffic lights is a twin.



Figure 2. Three kinds of digital copy. Solid arrows are automatic data flows. Dashed arrows are manual updates made by people.

The three side by side

	Digital model	Digital shadow	Digital twin
Data from city to copy	By hand, now and then	Automatic	Automatic
Results from copy to city	By hand	By hand	Automatic in the strict sense; in city practice a person often approves
How up to date	As of the last survey	Close to live	Close to live
Example	A 3D model from one aerial survey	A live traffic map	Traffic sensors, a queue forecast and retimed signals

Strictly, if a person must approve each change, the return flow is manual and the copy is a digital shadow. In city work the word twin is often used more loosely, for a copy whose results feed routinely into decisions with a person in the loop.

In practice most city projects mix all three. The buildings might be a digital model, refreshed once a year. The traffic layer might be a digital shadow. Only a few loops might be true twins. That is fine. The labels simply help you say clearly which part does what.

More than a 3D model

A beautiful 3D model on its own is a digital model. It may look impressive, yet it cannot tell you anything new about today. A dashboard of sensor numbers with no model of the place is limited too. It shows readings, but it cannot tell you what a new building or a closed road would do. A twin brings the shape of the city, live data and simulation together in one place.

> **SAY IT LIKE THIS**
> “A digital twin is a digital copy of a place that stays connected to it. Data comes in automatically, and what we learn goes back out into decisions.”

> **IN ONE LINE**
> A twin is a digital copy of the city that keeps in step with it and helps decide what to do.

2

A City in Six Layers

The parts every twin is built from.

It helps to picture a twin as a stack of layers. Each layer does one job and passes its results to the next. Real systems draw the lines in different places, but almost all of them have these six parts. One term first. The **Internet of Things**, or **IoT**, means everyday devices that measure something and send the result over a network.

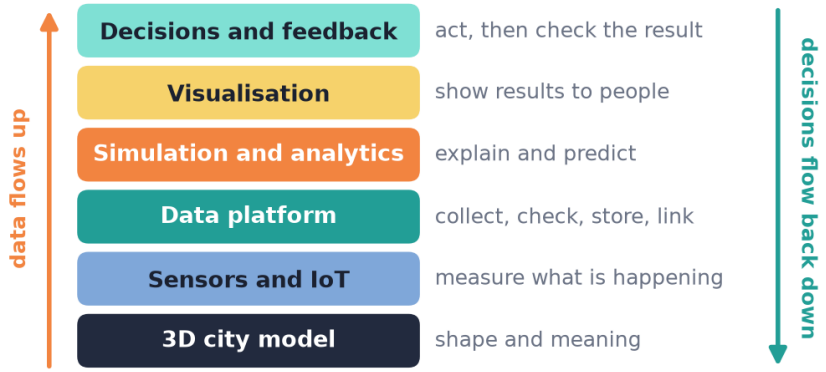


Figure 3. The six layers of an urban digital twin and the job each one does. Data flows up the stack, and decisions flow back down into the city.

A quick example shows the stack at work. The 3D model knows where a road and its junctions are. Traffic counters on that road send readings every minute. The platform checks each reading and links it to the right stretch of road. A traffic model predicts where queues will form. A dashboard shows the forecast, and an operator retimes the lights.

Data goes up, decisions come down

Read the stack from the bottom up and you follow the data. Read it from the top down and you follow the decisions. A decision made at the top, such as closing a road, changes the city. The sensors at the bottom then measure the effect, and the loop starts again. This loop is what sets a twin apart from a one-off study.

TIP

When you meet a new twin project, ask which of the six layers it really has. Some projects are strong on the 3D model and the visuals and thin on everything else.

IN ONE LINE

Every twin combines six layers, from the 3D model at the bottom to decisions at the top, joined in a loop.

3 The 3D City Model

The body of the twin, with its shape and its meaning.

The **3D city model** is the base layer. It describes the shape of the city, including the ground, buildings, roads, bridges, trees and water. A good city model also stores meaning. Each object knows what it is, such as a wall, a roof or a road lane, and carries facts like its height, age or use. Models that store this meaning are called **semantic** models.

Where the shapes come from

- **Aerial photos and laser scanning.** Planes or drones fly over the city. **Lidar**, which measures distance with laser pulses, records millions of points on roofs and the ground. Software turns the points into building shapes.
- **Building footprints.** The outline of each building on the ground, from maps or land records. Raising each footprint to a measured height gives a simple block model.
- **Design files.** New buildings are often designed with **BIM** (Building Information Modelling), where every part of a building is a digital object. These files can join the city model once the building is finished.
- **Street-level capture.** Cameras and scanners on cars or backpacks record facades and street furniture.

Levels of detail

Not every task needs every window. City models therefore come in **levels of detail**, usually shortened to **LoD**. The CityGML standard names them, and most tools follow the same idea.

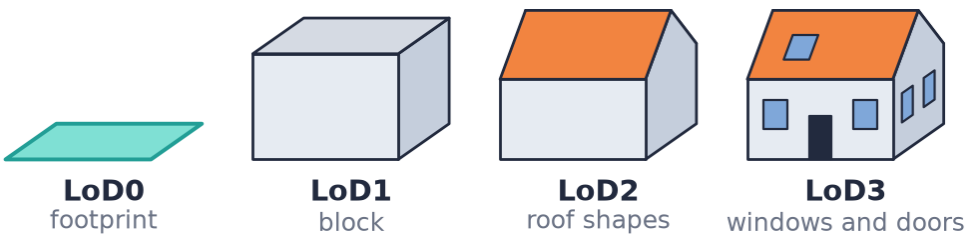


Figure 4. Four levels of detail for the same house. LoD0 is the footprint, LoD1 a simple block, LoD2 adds the roof shape and LoD3 adds windows and doors.

LoD1 and LoD2 are the workhorses of city twins. They are cheap enough to build for a whole city and detailed enough for many tasks. Facade detail at LoD3 is usually captured only for a few streets. Older versions of CityGML also had LoD4 for building interiors. Version 3.0 folds interiors into the other levels.

Which level for which job

Level	What it shows	Good for
LoD0	Footprints on the ground	2D maps, counting buildings, land use
LoD1	Blocks with one height each	City-wide 3D views, floor area estimates, first noise and wind studies
LoD2	Simplified roof shapes, with walls and roofs as separate surfaces	Solar potential of roofs, shadows, energy estimates
LoD3	Facades with windows, doors and balconies	Detailed street views, heritage records, facade studies

CityGML and CityJSON in one paragraph

CityGML (City Geography Markup Language) is an open standard from the **Open Geospatial Consortium** (OGC), the international body that agrees on standards for maps and location data. It defines what a city object is, such as a building, road or tree, and how its parts relate. Its usual file format is GML, a kind of XML. XML is a widely used text format for structured data. **CityJSON** stores the same kind of model in JSON, a lighter text format widely used on the web. Both keep the meaning of each object, so a twin can ask questions like “which buildings are taller than 20 m and were built before 1970?”

WATCH OUT

A 3D model can be precise and still out of date. Record when each part was captured, and plan how it will be refreshed.

IN ONE LINE

The 3D city model gives the twin its shape and its meaning, at a level of detail that fits the task.

4

Sensors and the Internet of Things

The senses of the twin, and why readings need checking.

Sensors let the twin feel what the city is doing. In a city they include traffic counters, weather stations, air quality monitors, water level gauges, smart energy meters and parking sensors. Together they form the Internet of Things layer of the twin.

What a reading contains

Each reading is small. It has a value, a unit, a time and a place, plus the name of the sensor that made it. A reading might say that sensor 17 measured 23.4 °C at 14:05 on a certain street corner. The twin's job is to attach that number to the right object in the 3D model.

Sensor	What it measures	Often reports
Traffic counter (road loop or camera)	Vehicles passing, speed	Every few seconds to every few minutes
Weather station	Air temperature, humidity, wind, rain	Every few minutes
Air quality monitor	Fine particles and gases	Every few minutes to every hour
Smart energy meter	Electricity or heat use	Every 15 to 60 minutes
Water level gauge	Level in a river or drain	Every few minutes, faster in storms
Satellite or drone images	Land cover, surface heat, change	Days to months

Real sensors are messy

Sensors drift, break, lose power and pick up odd values. Before a twin trusts a reading, it checks it. Figure 5 shows two days from one made-up temperature sensor, created by the script that builds this guide.

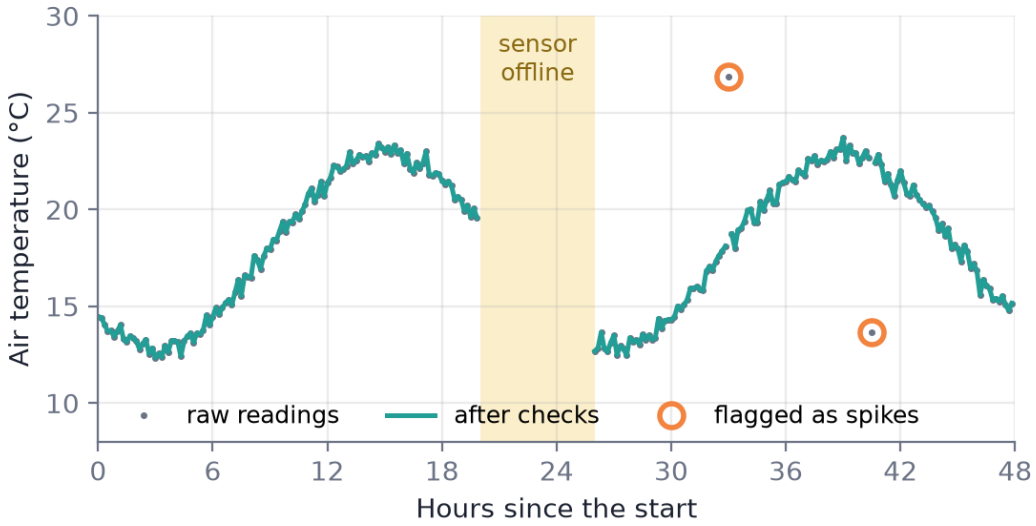


Figure 5. Two days of readings from one made-up street sensor, one every 10 minutes. Simple checks flag the spikes, and the gap is marked so nobody mistakes it for real data.

The script covers 288 slots of 10 minutes each. 36 of them are empty because the sensor went offline for 6 hours. The cleaning step flagged 2 readings as spikes, because each one sat more than 4 °C away from the median of the three readings before and after it. This is a form of the jump check below.

Three simple checks catch most problems.

- **Range check.** Is the value possible at all? An air temperature of 90 °C in a street is not.
- **Jump check.** Did the value change faster than nature allows? Compare it with the readings just before and after.
- **Neighbour check.** Does it agree with nearby sensors? One sensor far from all the others is suspect.

TIP

Store raw readings and cleaned readings separately. If a cleaning rule later turns out to be wrong, you can run it again on the raw data.

IN ONE LINE

Sensors give the twin its senses, and every reading needs a time, a place and a quality check.

5 The Data Platform

The plumbing that joins everything up.

The **data platform** joins the other layers together. It collects data from many sources, checks it, stores it and serves it to other software.

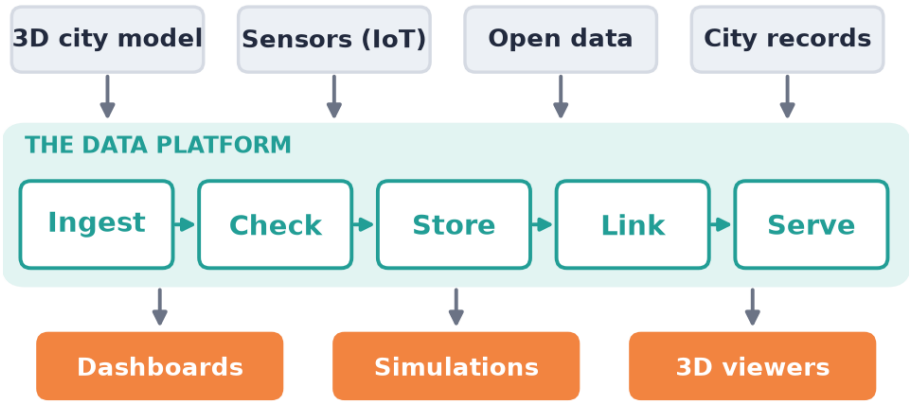


Figure 6. The data platform takes data from many sources, does five jobs in order, and serves the results to the apps people use.

The five jobs of the platform

1. **Ingest.** Receive data as it arrives. Sensors often send messages with **MQTT**, a light messaging system built for small devices.
2. **Check.** Apply quality checks like those in Chapter 4, and log every change.
3. **Store.** Readings go in a **time series database**, built for long lists of time-stamped values. Shapes go in a spatial database.
4. **Link.** Join everything by shared **identifiers**, or IDs. A meter reading is useful only if the platform knows which building the meter belongs to.
5. **Serve.** Offer the data through an **API** (application programming interface), a set of agreed requests that other programs can send.

Why linking is the hard part

City departments often keep separate lists of buildings, streets and assets, each with its own codes. One office knows a building by its address, another by its land parcel and a third by its meter number. A twin works only when these lists can be matched, and that slow work often decides whether it succeeds.

> **IN ONE LINE**
> The platform ingests, checks, stores, links and serves data, and linking by ID is the hardest of the five jobs.

6 Simulation and Analytics

Explaining the past and predicting what comes next.

A twin that only shows data is a mirror. Simulation and analytics turn it into a tool for thinking. **Simulation** means running a model of how something works, such as traffic, heat or water, to see what it produces. **Analytics** means finding patterns in data, for example with statistics or machine learning.

Three kinds of model

Kind	How it works	Example
Physics-based	Uses rules about how the world works, such as water flowing downhill or heat moving from warm to cold	A flood model that routes rain over the ground and into drains
Data-driven	Learns patterns from past data	A model that predicts tomorrow's energy use from the weather forecast
Hybrid	Combines rules with learning from data	A traffic model whose settings are tuned from live counts

Calibration makes the model match the city

A model has settings, called **parameters**. When it is first built, those settings are educated guesses. **Calibration** means adjusting them until the model matches measured data. A twin is a good place to calibrate, because new measurements arrive all the time.

Figure 7 shows a small example computed for this guide. A made-up building's daily energy use depends on how cold it is outside. The first guess is clearly off. Fitting two parameters to 60 days of meter readings brings the typical error down from 30 to 11 kWh per day. On 60 new made-up days that it was not tuned on, the error is 10 kWh per day.

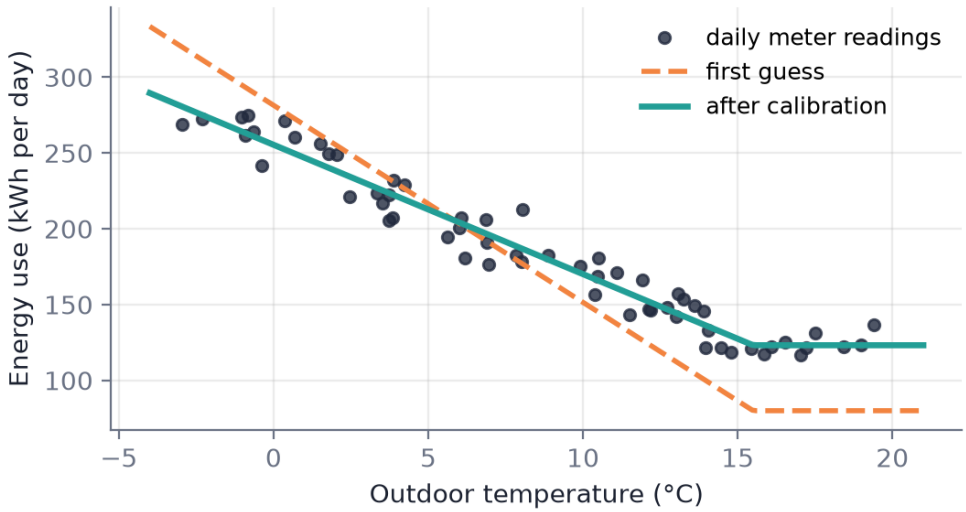


Figure 7. Calibrating a simple energy model of one made-up building. Each dot is one day. Computed for this guide.

The model is simple. The building uses a fixed base amount each day, plus extra heating for every degree below a chosen base temperature, here 15.5 °C. Calibration found a base of 123 kWh and 8.5 kWh for each degree. Real models have many more settings, but the idea is the same. Here the “typical error” is the **root mean square error** (RMSE), a standard way to average the size of the misses.

WATCH OUT

A model that matches the past can still fail on the future. Always test it on data it was not tuned on, such as a different month.

IN ONE LINE

Simulation and analytics turn the twin's data into explanations and predictions, and calibration keeps them honest.

7 Seeing, Deciding, Acting

Visualisation, decisions and the feedback loop.

The top two layers of a twin are about people. **Visualisation** shows results so that people can understand them quickly. Decisions and feedback turn understanding into action, and then check whether the action worked.

Many ways to see a twin

- **3D web maps** show the city in a browser, with layers you can switch on and off.
- **Dashboards** give charts and alerts to people who run services every day.
- **Game engines** and **virtual reality** draw realistic scenes for consultation.
- **Plain maps and reports** still matter. Many decisions are made in meetings.

Closing the loop

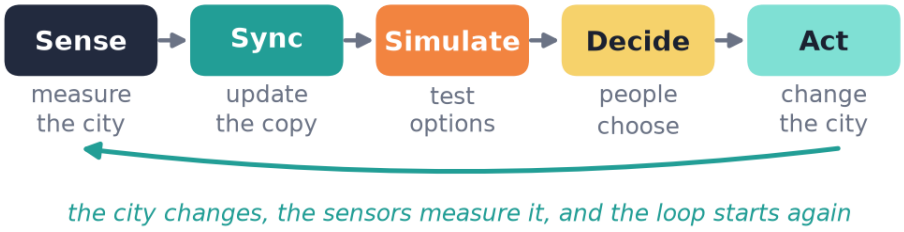


Figure 8. The twin loop. Each turn of the loop measures the result of the last decision.

The twin senses the city, syncs its copy, simulates options and helps people decide. The decision is acted on, and the sensors measure what changed. A short loop, such as adjusting traffic signals, may turn every few minutes. A long loop, such as a new planning rule, may take years to close.

Who decides?

In many city twins, the twin recommends and a person approves, which strictly makes the return flow manual (see Chapter 1). Fully automatic control is kept for narrow, well-tested tasks, because someone must be accountable for each decision that affects people's lives.

SAY IT LIKE THIS

“The twin suggests options and shows the trade-offs. People make the decision, and the twin then checks whether it worked.”

IN ONE LINE

A twin helps when its results reach people who can act, and it checks the outcome.

8

How Fresh Is the Data?

Update rates, and why they matter.

Different parts of a city change at different speeds. A building stands for decades. A traffic queue can form in a minute. A twin should refresh each layer about as fast as the decisions that depend on it. Refreshing faster costs money, energy and storage. Refreshing slower means decisions use old information.

What changes	How fast it changes	A sensible refresh
Ground shape and buildings	Years	Yearly, or when a building is finished
Land use and trees	Months to years	Yearly or each season
Energy use	Hours	Every 15 to 60 minutes
Air temperature and air quality	Minutes to hours	Every few minutes
Traffic flow	Seconds to minutes	Every few seconds to every minute
Water levels in a storm	Minutes	Every few minutes, faster during alerts

Latency is the delay between something happening and the twin showing it. It includes the time to measure, send, check and process the data. Averaging adds latency too, because an average can only be reported once its time window has ended.

Worked example: spotting a traffic incident

Figure 9 shows a made-up morning on one road, computed by this guide's script. Traffic flows at about 30 vehicles per minute. At 08:12 a crash slows it to about 12 vehicles per minute until 08:50. The twin raises an alert when the reported flow drops below 20 vehicles per minute. The same traffic is reported three ways, every minute, as 15-minute averages and as hourly averages.

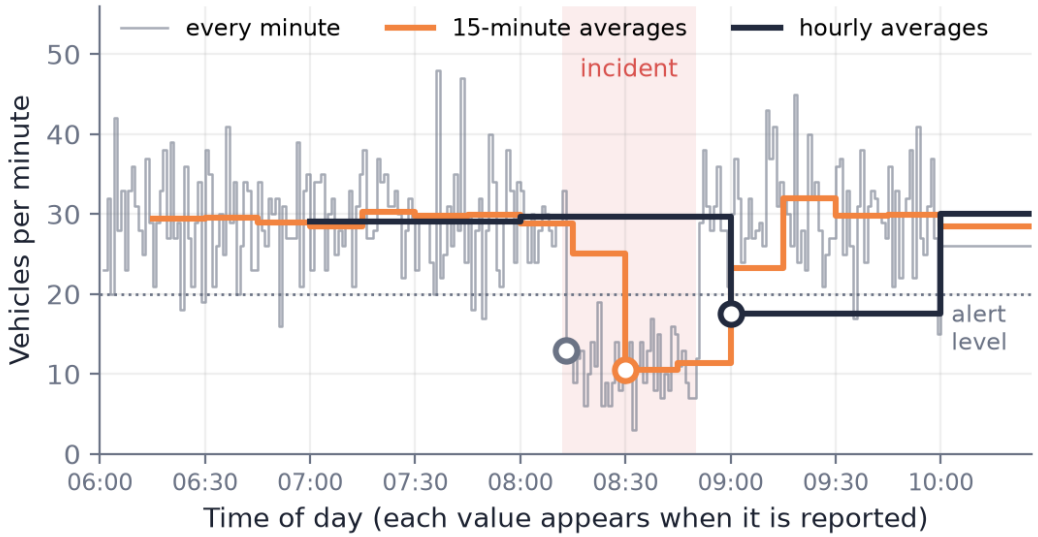


Figure 9. One made-up morning, reported at three different rates. Circles mark the first report below the alert level after the incident began. Computed for this guide.

Reporting and alert rule	First alert after the crash	Delay after the incident began	False alarms before the crash
Every minute, one low report	08:13	1 minute	5
Every minute, three low reports in a row	08:15	3 minutes	0
15-minute averages	08:30	18 minutes	0
Hourly averages	09:00	48 minutes	0

The every-minute feed is noisy. Before the crash it dipped below the alert level 5 times, and each dip would have been a false alarm. Many alert systems therefore wait for several low reports in a row. With a rule of three in a row, the alert comes at 08:15, 3 minutes after the crash, and none of the earlier dips sets it off.

The hourly averages are accurate. They simply arrive too late for this decision. For a yearly report on traffic volumes, hourly data would be perfectly fine, and far cheaper to store.

TIP

For each layer, write down the fastest decision that depends on it. Set the refresh rate from that, and add a margin.

IN ONE LINE

Refresh each layer as fast as the decisions that rely on it, and no faster.

9 Maturity Levels

From a static map to a twin that helps run the city.

Twins grow in steps. Several organisations have published maturity scales, and they differ in the details. Most follow the same path, from a static model to a connected one, then to prediction, and finally to feedback into the city. The ladder below combines these common ideas.

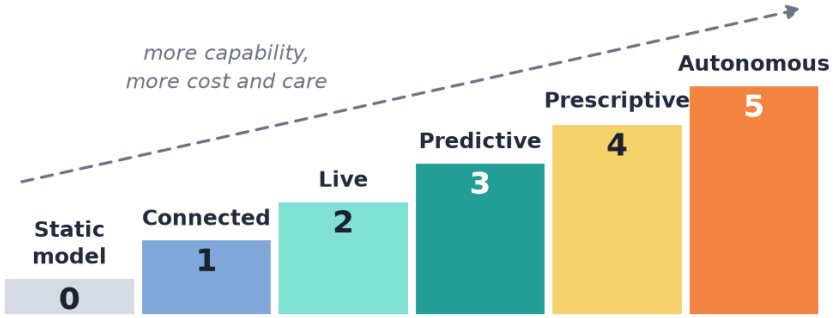


Figure 10. A simple maturity ladder for urban digital twins. Each step adds capability, and each step costs more to build and run.

Level	Name	What it can do	Question it answers
0	Static model	A 2D or 3D model, updated by hand	What is there?
1	Connected	The model is linked to records and data by ID	What do we know about it?
2	Live	Sensor data flows in automatically	What is happening now?
3	Predictive	Simulations and forecasts run on live data	What will happen next?
4	Prescriptive	Compares options and recommends actions	What should we do?
5	Autonomous	Acts on its own within strict limits	Can it safely handle this alone?

Higher is not always better. A city may need a level 1 twin for planning and a level 3 twin for flooding. Pick the level that matches the decision you need to support. Be wary of claims of level 5. Fully automatic control of city systems is rare, and it belongs only in narrow tasks with clear safety rules and a person who can step in.

IN ONE LINE

Maturity levels describe how far a twin goes from showing the city to helping run it, and each step should earn its cost.

10

Standards in Plain Words

The shared languages that let the parts work together.

A **standard** is a published agreement about how to describe or exchange something. Standards let a model made in one program open in another. They also protect a city from **lock-in**, which is being stuck with one supplier because the data cannot easily move. Four standards come up again and again in city twins.

Standard	Looked after by	What it carries	Think of it as
CityGML and **CityJSON**	OGC	City objects such as buildings, roads and trees, with their meaning	The city's dictionary
IFC (Industry Foundation Classes)	buildingSMART; also published as ISO 16739	One building or structure in full detail, including walls, slabs, rooms and services	The building's blueprint
SensorThings API	OGC	Sensors, what they measure, and their readings over time	The city's diary of measurements
3D Tiles	OGC (first developed by the company Cesium)	Huge 3D scenes cut into tiles for fast streaming	A map that streams in a little at a time, getting sharper as you zoom in

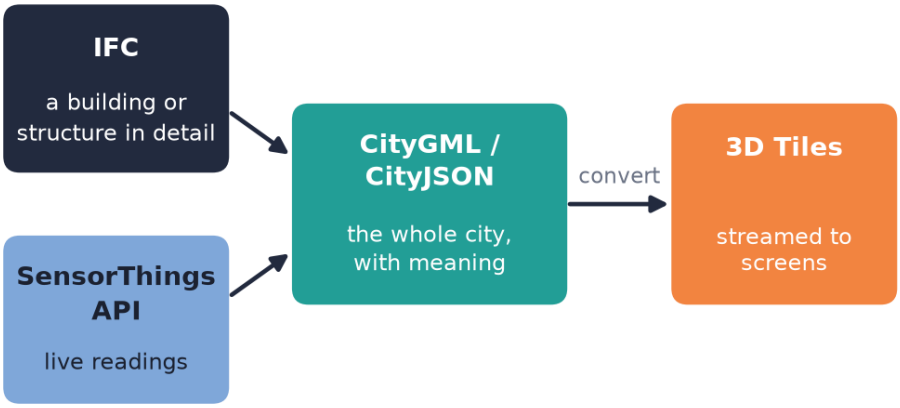


Figure 11. How the four standards fit together. Building detail and live readings, linked by ID, join the city model, which is converted into tiles for fast viewing.

A few more names you will meet

- **glTF.** A compact format for 3D shapes and textures, sometimes called the “JPEG of 3D”. 3D Tiles use it inside most tiles.
- **GeoJSON.** A simple JSON format for points, lines and areas on a map.

- **NGSI-LD.** A standard API, published by ETSI (the European Telecommunications Standards Institute), for sharing information about things, such as whether a parking space is free. FIWARE, an open-source toolkit for smart city platforms, uses it alongside the older NGSI-v2.
- **MQTT.** The light messaging system from Chapter 5 that many sensors use to send readings.

Joining building and city

IFC and CityGML grew up in different worlds. IFC comes from architecture and construction, where models are detailed and cover one site. CityGML comes from mapping, where models are simpler and cover large areas. Converting between them is possible but rarely perfect. Detail is lost going from IFC to CityGML, and missing information must be filled in going the other way. Plan for some manual work.

> **TIP**
> Ask any supplier which open standards their twin can import and export. If the answer is “only our own format”, your data may be hard to move later.

> **IN ONE LINE**
> Open standards are the shared languages of the twin, and they protect the city from lock-in.

11 What Twins Are Used For

Mobility, energy, heat, flooding and planning.

A twin is worth building only if it helps answer real questions. The five areas below are among the most common. Each needs a different mix of data, models and update speed, as Figure 12 shows.

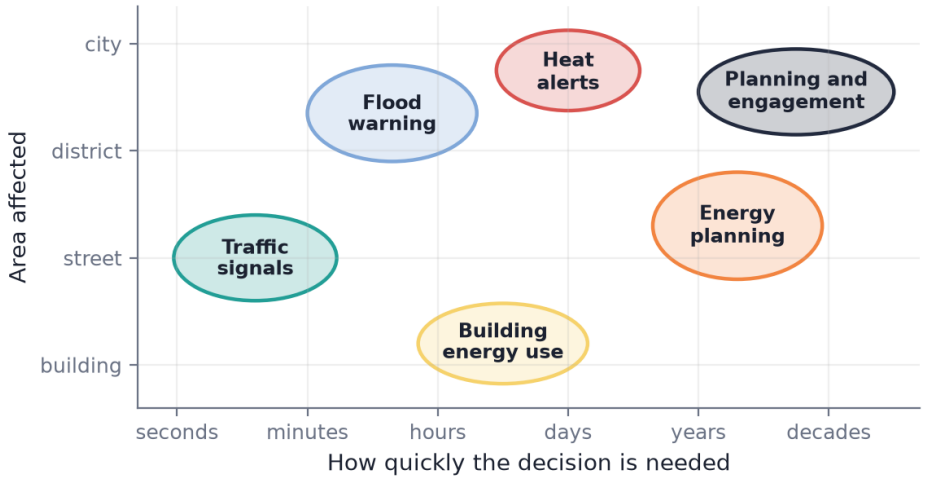


Figure 12. Typical uses of a twin, placed by how quickly a decision is needed and how large an area it affects. The placement is rough and meant only as a guide.

Mobility

Traffic and transport twins combine road networks, public transport timetables and live counts. They help retime traffic signals, plan diversions during roadworks and test new bus routes before they run.

Energy

Energy twins estimate how much heating, cooling and electricity buildings need. They use building age, size, use and roof shape from the 3D model. Cities use them to see which areas suit shared heating networks, which roofs suit solar panels and where insulation would help most.

Urban heat

Streets with little shade and many dark surfaces get hot. Heat twins combine building shapes, trees, surface materials and weather data. They help find hot spots and test cooling ideas such as trees, lighter surfaces and shade.

Flooding

Flood twins combine ground height, drains, rivers and rainfall. Before a storm they forecast which streets may flood. During a storm they compare forecasts with water level sensors. Afterwards they help test new drains, green spaces and barriers.

Planning and public engagement

Planners use the twin to see a proposed building in context, including its height, its shadows and the views it blocks. A clear 3D view helps residents understand a proposal and comment on it, without needing to read technical drawings.

Well-known examples

Virtual Singapore is a detailed 3D model of the whole city-state, built with a shared data platform for planning, research and testing ideas. **Helsinki** has published 3D models of the city as open data, including a semantic city model and a photo-realistic mesh. Many cities share 3D building data openly, so anyone can start experimenting.

> **SAY IT LIKE THIS**
> “Which decision will this twin help us make, and who will make it?”

> **IN ONE LINE**
> Twins earn their keep in mobility, energy, heat, flooding and planning, each with its own data and its own pace.

12

Designing with What-If Scenarios

Testing ideas in the copy before building them in the city.

A **scenario** is a possible future described in enough detail to model it. “What if half the roofs were green?” is a scenario. Designing with a twin means building several scenarios, running the same models on each and comparing the results side by side. The **baseline** is the scenario that describes the city as it is today.

The recipe

1. Set up the baseline, and check that the model matches measurements.
2. Change one thing at a time, such as a new building, more trees or a closed road.
3. Run the same model with the same weather or demand on every scenario.
4. Compare results using measures agreed in advance, such as flooded area, peak temperature or travel time.
5. Combine the best ideas and test the combination, because effects do not always add up neatly.

Example: one storm on one block

The script that builds this guide models a made-up block of 10,000 m² during a storm that drops 40 mm of rain. That is 400 m³ of water in total. It uses a simple bucket rule. Each kind of surface sends a fixed share of the rain into the drains and keeps the rest. The shares below are assumptions chosen for illustration. Real studies use detailed drainage models.

Surface	Share of the block	Share of rain that runs off today	Change tested
Roofs	40%	90%	Green roofs on half of them, 50% runs off
Roads	25%	90%	None
Pavements	15%	85%	Permeable paving, 30% runs off
Gardens and verges	20%	20%	None

Figure 13 shows the results. In this toy, green roofs on half the roofs cut runoff by 32 m³. Permeable pavements cut it by 33 m³. Both together cut it by 65 m³, which is 22% less than the baseline. With this simple rule the savings add up exactly. In a real drainage model they often do not, which is why step 5 of the recipe matters.

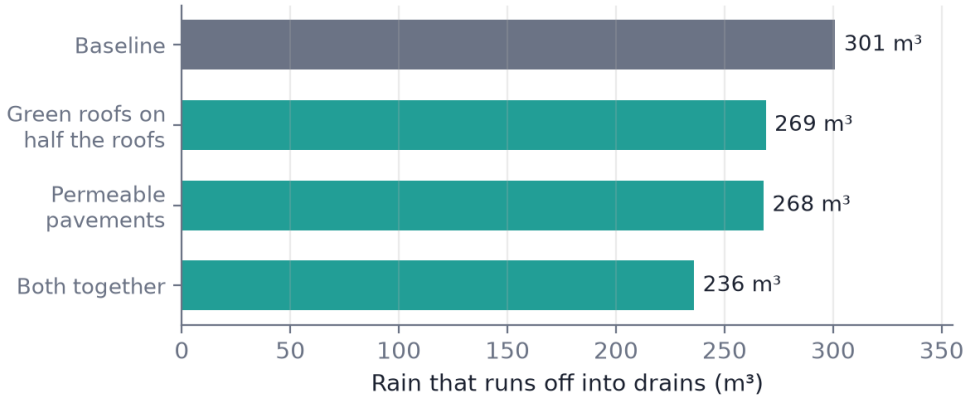


Figure 13. Rain running off the made-up block in four scenarios. Computed for this guide with the simple bucket rule.

WATCH OUT

A scenario result is only as good as the model and its inputs. Show the assumptions next to the result, and say how uncertain it is.

IN ONE LINE

Designing with a twin means changing one thing at a time in the copy, running the same tests and comparing fairly.

13

A Mini Twin of One Street

A tiny twin built in code, step by step.

This chapter builds a very small twin in the Python script that makes this guide. Everything is made up, but each step mirrors a real twin. The street is 160 m long and 16 m wide from facade to facade, with 8 buildings on each side.

Step 1. A model of the street

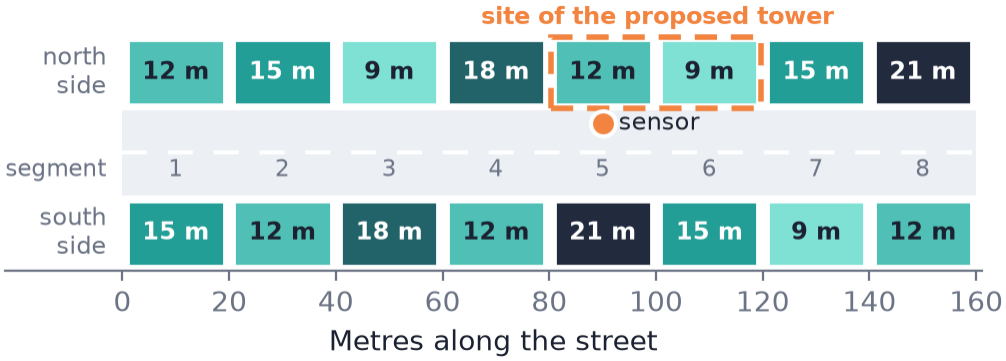


Figure 14. The mini twin seen from above. Each block is a building with its height. The dot is the temperature sensor, and the dashed outline marks a proposed tower.

Each building has a footprint and a height. From these the script computes two measures that matter for heat and daylight. The **height-to-width ratio** is the average building height divided by the street width. The **sky view factor** is the share of the sky you can see from the middle of the street, from 0 (no sky) to 1 (open sky). For a long, straight street a standard formula gives it from the street width and the height of each side. The script treats each segment as part of an endlessly long street, so the result is a rough guide.

The script finds an average building height of 14.1 m, an average height-to-width ratio of 0.88 and an average sky view factor of 0.51. The most enclosed parts of the street are segments 5 and 8, each with a sky view factor of 0.46.

Step 2. A sensor keeps the twin honest

A temperature sensor on a lamp post in segment 5 sends a reading every hour for two days. A citywide weather model also predicts the temperature, but it knows nothing about this street, which runs warmer.

The twin learns the difference between the readings and the model, called the **bias**, and adds it to the model. Each new reading moves the bias 40% of the way towards the latest difference, so a single odd reading moves it only part of the way. In practice the spike checks from Chapter 4 run first. When the sensor goes offline for 8 hours, the twin keeps using the last bias it learned.

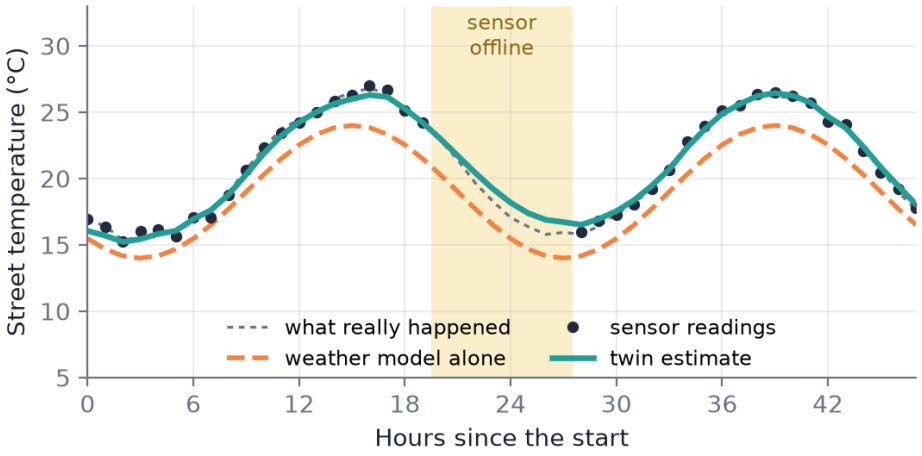


Figure 15. Two days in the mini twin. The twin estimate is the weather model plus a bias learned from the sensor. Computed for this guide.

Over the two days, the weather model alone has a typical error (RMSE, from Chapter 6) of 2.0 °C. The twin cuts this to 0.5 °C. While the sensor is offline the twin's typical error is 0.8 °C, against 2.0 °C for the model alone. Real twins use many sensors and smarter blending, but the logic is the same.

Step 3. A what-if test

A developer proposes replacing two low buildings on the north side, in segments 5 and 6, with one 36 m tower. The twin recomputes the sky view factor along the street.

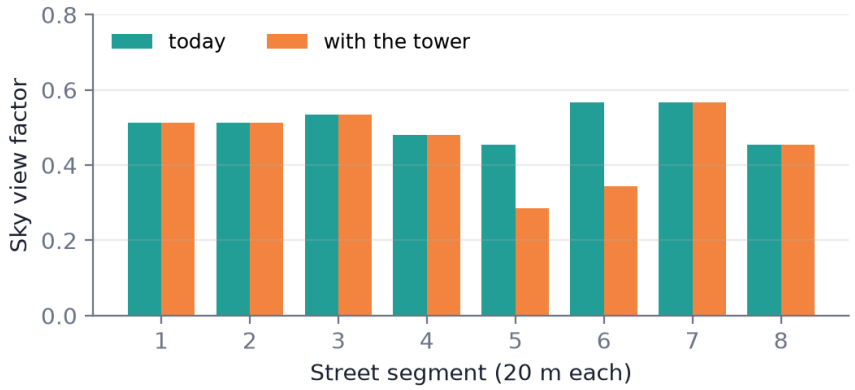


Figure 16. Sky view factor along the street, today and with the proposed tower. Computed for this guide with the long-street formula.

In segment 5 the sky view factor drops from 0.46 to 0.29, and in segment 6 from 0.57 to 0.34. Less open sky means less daylight in the street and slower cooling at night. The twin cannot say whether the tower is a good idea. It shows one trade-off clearly, so people can weigh it against the homes or jobs the tower would bring.

IN ONE LINE

Even a tiny twin shows the whole idea, with a model of the place, live data that corrects it and what-if tests that inform decisions.

14

Governance, Privacy and Cost

The questions that decide whether a twin lasts.

Besides its technology, a twin needs agreement on who runs it, who may use it and who pays for it. **Governance** is the name for these rules and responsibilities.

Governance questions to answer early

Question	Why it matters
Who owns the twin and each layer?	Without an owner, data goes stale and errors stay.
Who may see and use which data?	Some data is public, some sensitive or personal.
Who answers for the decisions?	Any model can be wrong. Someone must be accountable.
How long is data kept?	Old data costs money and can put privacy at risk.
How is quality checked and shared?	Users must know how far to trust each layer.

Privacy

Some sensors can capture faces, number plates or phone signals. A twin rarely needs to identify people. Counts, flows and averages are enough.

- **Collect less.** Measure only what a decision needs.
- **Aggregate early. Aggregation** turns individual detections into totals, such as counts per street per quarter hour. Do it as close to the sensor as possible.
- **Anonymise with care.** Removing names is not enough when movement patterns can still point to one person.
- **Be open.** Tell residents what is measured, why, and for how long, and follow the data protection laws that apply where you are.

Cost

The visible cost is building the model and buying sensors. Running costs then continue every year. Sensors need power, repair and replacement. The model needs refreshing. Storage, computing, licences and skilled staff all need paying for. Budget for running the twin over its whole life, as well as for building it.

> **IN ONE LINE**
> A twin lasts when ownership, privacy and running costs are agreed as carefully as the technology.

15

Common Pitfalls

Mistakes that are easy to make and easy to avoid.

Twin projects tend to stumble in familiar ways. Start with the question the twin should answer, then choose the technology. The table lists common traps, with a simple fix for each.

Pitfall	What it looks like	What to do instead
The pretty picture	A striking 3D scene with no live data and no analysis	Start from a decision, then build only the layers it needs
Sensor sprawl	Thousands of sensors, little use of their data	Add sensors where a model or a decision needs them
The stale model	The 3D model was captured once and never updated	Plan refresh cycles and record capture dates
Lock-in	Data lives in one supplier's closed format	Require open standards for import and export
The black box	Results with no explanation of models or assumptions	Publish assumptions, model versions and known limits
False precision	Results shown to many decimal places	Show ranges and say how uncertain they are
No owner	A pilot ends and nobody maintains the twin	Name an owner and budget for running costs
Forgetting people	Residents and staff are not involved	Design with the people who will use the twin and be affected by it

A checklist before you start

1. Which decision will the twin support, and who makes it?
2. Which of the six layers does that decision really need?
3. How fresh must each layer be?
4. Which open standards will you use for each kind of data?
5. What personal data could be captured, and how will you avoid or protect it?
6. Who owns the twin, and who pays to keep it running?
7. How will you know whether the twin helped?

IN ONE LINE

Start from a decision, keep the data fresh and open, show uncertainty and name an owner.

16

Try It Yourself

Free tools, a short recipe and an eight-week plan.

You can build a small twin on an ordinary laptop with free tools. The table lists widely used options. Most are open source, which means anyone may use, study and change them.

Tool	What it does	Good for
QGIS	Free desktop mapping software, with a 3D map view	Exploring data, preparing footprints and heights
PostGIS	Adds spatial data (points, lines and areas) and spatial queries to the PostgreSQL database	Storing and querying spatial data
3DCityDB	Free database and tools for CityGML city models	Managing a semantic 3D city model
CesiumJS	Open-source library for 3D globes and maps in a web browser	Web viewers that stream 3D Tiles
Blender	Free 3D modelling and rendering software	Editing shapes, making images and videos
Unreal Engine, **Unity**	Game engines, free for personal and small-scale use under their own licences (not open source), that can show map data through plugins	Realistic, interactive scenes
FIWARE	Open-source building blocks for smart city data, using the NGSI family of APIs	Managing live data about city objects
FROST-Server	Open-source server for the SensorThings API	Storing and serving sensor readings
Grafana	Open-source dashboards for time series data	Live charts and alerts
Eclipse SUMO	Open-source traffic simulation	Testing traffic scenarios
EnergyPlus	Open-source building energy simulation	Detailed energy studies of buildings

A recipe for your first mini twin

1. Pick a small area you know well, such as one street or one block.
2. Download building footprints from OpenStreetMap or a city open data portal. Add heights where they exist.
3. Load them into QGIS and raise them into a simple LoD1 block model.

4 Find one open data feed for the area, such as weather or traffic counts.

5 Store the readings with a time and a place, and link each one to a street or building ID.

6 Compute one useful measure, such as the height-to-width ratio of the street.

7 Make one what-if change, such as a taller building, and compare before and after.

8 Show the result on one clear page, and write down every assumption you made.

An eight-week plan

Week	Focus	Do this
1	The big idea	Read Chapters 1 and 2. Find two public city twin projects and list which of the six layers each one has.
2	3D city models	Download open 3D building data for any city. Open it in QGIS or a web viewer and look at its level of detail.
3	Sensors and data	Find an open sensor feed. Plot two days of readings and mark any gaps and spikes.
4	Linking	Join the readings to streets or buildings, by ID or by location. Note every mismatch you find.
5	Simple models	Build one simple model, such as energy use against outdoor temperature, and calibrate it.
6	Viewing	Show your model and data in a 3D web viewer or a simple dashboard.
7	What-if scenarios	Test two or three scenarios and compare them with measures you chose in advance.
8	Share and reflect	Write one page on what your twin can and cannot tell you, and show it to someone new to the topic.

IN ONE LINE

Start small, use open tools, and build one complete loop before making anything bigger.

Myths and Facts

Common misunderstandings, cleared up.

Myth	Fact
A digital twin is just a very detailed 3D model.	The 3D model is only the base layer. A twin also needs live data, analysis and a link back to decisions.
More sensors always make a better twin.	Sensors help only when their data is checked, linked and used. A few well-placed sensors can beat many idle ones.
A twin must look photo-realistic.	Many useful twins use simple blocks and plain charts. The right level of detail depends on the question.
A twin replaces the judgement of planners and engineers.	A twin informs decisions. People still weigh the trade-offs and stay accountable for the choice.
Once it is built, a twin is finished.	A twin needs constant care. Without fresh data it slowly turns back into a static model.
Only big, wealthy cities can have a twin.	A small town can start with open data, free tools and one clear question.

Quick Quiz

Ten questions. Answers are on the next page.

1 What is the difference between a digital model, a digital shadow and a digital twin?
2 Name the six layers of a typical urban digital twin.
3 What does LoD2 add compared with LoD1?
4 Why must every sensor reading carry a sensor ID or a location?
5 What is calibration?
6 In the traffic example, why did the hourly feed raise the alert so late?
7 Which standard would you use to share live sensor readings?
8 What is a scenario, and what is the baseline?
9 In the mini twin, how did the twin stay accurate while the sensor was offline?
10 Give two ways to protect privacy in a city twin.

Quiz Answers

No peeking until you have tried.

1. A digital model has no automatic link. A digital shadow gets data automatically from the city, but nothing flows back on its own. A digital twin has automatic flows in both directions.
2. 3D city model, sensors and IoT, data platform, simulation and analytics, visualisation, and decisions and feedback.
3. The shape of the roof. LoD1 is a flat-topped block.
4. So that the platform can link it to the right building or street in the model.
5. Adjusting a model's parameters until its results match measured data.
6. An hourly average is reported only when the hour ends, and averaging softens a short drop. The alert came 48 minutes after the incident began.
7. The OGC SensorThings API.
8. A scenario is a possible future described in enough detail to model. The baseline is the scenario that describes the city as it is today.
9. It kept using the bias it had learned from earlier readings and added it to the weather model.
10. Any two of these: collect only what a decision needs, aggregate early into counts, anonymise with care, and tell residents what is measured and why.

Glossary

Plain definitions of the words in this guide.

Term	Meaning
3D city model	A digital model of the ground, buildings, roads, trees and water of a city, often storing what each object is.
3D Tiles	An open standard for streaming very large 3D scenes in pieces, so they load quickly.
Aggregation	Turning individual records into totals or averages, such as counts per street per quarter hour.
Analytics	Finding patterns in data, for example with statistics or machine learning.
API	Application programming interface. A set of agreed requests that one program can send to another.
Baseline	The scenario that describes the city as it is today, used as the point of comparison.
Bias	A steady difference between a model and reality, such as a model that always runs 2 °C too cool.
BIM	Building Information Modelling. Designing a building as a set of digital objects with properties.
Calibration	Adjusting a model's parameters until it matches measured data.
CityGML	An open OGC standard for describing city objects and their meaning.
CityJSON	A JSON encoding of the CityGML data model, lighter and easier to use on the web.
Data platform	The layer that ingests, checks, stores, links and serves the twin's data.
Digital model	A digital copy with no automatic link to the real thing.
Digital shadow	A digital copy that receives data from the real thing automatically, with nothing flowing back on its own.
Digital twin	A digital copy linked to the real thing by automatic data flows in both directions.
Governance	The rules and responsibilities for who owns, uses, pays for and is accountable for the twin.
Height-to-width ratio	Average building height along a street divided by the street's width.

Term	Meaning
Identifier (ID)	A code that names one object, such as a building or a sensor, so that records about it can be joined.
IFC	Industry Foundation Classes. An open standard for detailed models of buildings and infrastructure, published as ISO 16739.
IoT	Internet of Things. Everyday devices that measure something and send the result over a network.
Latency	The delay between something happening and the twin showing it.
Level of detail (LoD)	How much detail a 3D city model holds, from footprints (LoD0) to facades (LoD3).
Lidar	A way of measuring distance with laser pulses, used to capture the shape of the ground and buildings.
Lock-in	Being stuck with one supplier because the data cannot easily move elsewhere.
MQTT	A light messaging system that many sensors use to send their readings.
OGC	Open Geospatial Consortium. The international body that agrees on open standards for maps and location data.
Parameter	A setting inside a model that can be adjusted, such as heating per degree of cold.
RMSE	Root mean square error. A standard way to measure the typical size of a model's errors.
Scenario	A possible future described in enough detail to model it.
Semantic model	A model where each object knows what it is, such as a roof or a road, and carries facts.
SensorThings API	An open OGC standard for sharing sensors and their readings over the web.
Simulation	Running a model of how something works to see what it produces.
Sky view factor	The share of the sky visible from a point, from 0 (none) to 1 (open sky).
Standard	A published agreement about how to describe or exchange something, so that different programs can work together.
Time series database	A database built for long lists of time-stamped values, such as sensor readings.
Urban digital twin	A digital copy of a city, or part of one, that stays linked to the real place by data flows.
Visualisation	Showing data and results as maps, charts or 3D scenes so that people can understand them quickly.

One-Page Cheat Sheet

Pin it next to your screen.

Idea	In short
Digital twin	A digital copy of a place, linked to it by automatic data flows in both directions.
Model, shadow, twin	No automatic link, a one-way link, a two-way link.
Six layers	3D model, sensors, data platform, simulation, visualisation, decisions and feedback.
Levels of detail	LoD0 footprint, LoD1 block, LoD2 roof shapes, LoD3 windows and doors.
A sensor reading	Value, unit, time, place and sensor ID, plus a quality check.
Platform jobs	Ingest, check, store, link, serve. Linking by ID is the hardest.
Calibration	Tune the parameters until the model matches measurements, then test on new data.
Refresh rate	As fast as the decisions that depend on the layer, and no faster.
Maturity	Static, connected, live, predictive, prescriptive, autonomous.
Standards	CityGML or CityJSON for the city, IFC for buildings and structures, SensorThings for readings, 3D Tiles for streaming.
What-if	Baseline first, change one thing, run the same tests, compare fairly.
Governance	Owner, access rules, accountability, how long data is kept, quality reporting.
Privacy	Collect less, aggregate early, anonymise with care, be open.
First step	Start from one decision and the people who make it.

Further Reading

Where the ideas come from.

Standards are revised from time to time, so check for the latest version before citing one.

- Kritzinger, W., Karner, M., Traar, G., Henjes, J. and Sihn, W. (2018). Digital twin in manufacturing: a categorical literature review and classification. *IFAC-PapersOnLine*, 51(11). The widely cited source of the model, shadow and twin classification.
- Grieves, M. and Vickers, J. (2017). Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems. In Kahlen, F.-J., Flumerfelt, S. and Alves, A. (eds), *Transdisciplinary Perspectives on Complex Systems*, pp. 85–113. Springer.
- Batty, M. (2018). Digital twins. *Environment and Planning B: Urban Analytics and City Science*, 45(5). A short, thoughtful editorial on twins for cities.
- Batty, M. (2013). *The New Science of Cities*. MIT Press. A classic on cities as systems of flows and networks.
- Ketzler, B., Naserentin, V., Latino, F., Zangelidis, C., Thuvander, L. and Logg, A. (2020). Digital twins for cities: a state of the art review. *Built Environment*, 46(4).
- Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S. and Çöltekin, A. (2015). Applications of 3D city models: state of the art review. *ISPRS International Journal of Geo-Information*, 4(4).
- Biljecki, F., Ledoux, H., Stoter, J. and Zhao, J. (2014). Formalisation of the level of detail in 3D city modelling. *Computers, Environment and Urban Systems*, 48.
- Kitchin, R. (2014). The real-time city? Big data and smart urbanism. *GeoJournal*, 79(1).
- Oke, T. R., Mills, G., Christen, A. and Voogt, J. A. (2017). *Urban Climates*. Cambridge University Press. Covers street canyons, the height-to-width ratio and the sky view factor.
- Open Geospatial Consortium (2021). *OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard*, version 3.0.
- Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A. and Vitalis, S. (2019). CityJSON: a compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards*, 4, 4.
- ISO 16739-1:2024. *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries. Part 1: Data schema*.
- Open Geospatial Consortium (2021). *OGC SensorThings API Part 1: Sensing*, version 1.1.
- Open Geospatial Consortium (2019). *3D Tiles Specification 1.0*. OGC Community Standard (version 1.1 followed in 2023).