1) Intro & Definition of Agile

- **Agile** = a mindset + lightweight methods to deliver **value early and often**, learning from feedback.
- Focus on **people & interactions**, **working product**, **customer collaboration**, **responding to change**.
- Output is measured by **outcomes** (value), not just documents or plans.

2) How is Agile different?

- **Iterative & incremental** vs. “big-bang” waterfall.
- **Plan lightly, adjust frequently** using real user feedback.
- Cross-functional teams, minimal handoffs, **continuous delivery** mindset.

3) Principles & a bit of history

- Born from the **2001 Agile Manifesto** (17 practitioners in Snowbird).
- 12 principles (e.g., satisfy customer early, welcome change, deliver frequently, sustainable pace, technical excellence).
- Many frameworks: **Scrum, Kanban, XP, Lean**, etc.

4) FAQs about Agile

- **Is Agile = Scrum?** No—Scrum is one Agile **framework**.
- **Documentation?** Yes—**just enough** to be useful.
- **Fixed scope?** Scope is flexible; **time & quality** are protected.
- **Works outside IT?** Yes (marketing, operations, HR, events).

5) Key Agile concepts

- **Backlog** (ordered list of work by value).
- **User Story**: *As a <user>, I want <capability> so that <benefit>.*
  Example (Android): “As a shopper, I want biometric login so that sign-in is quick.”
- **MVP**: smallest slice that delivers value.
- **Definition of Done (DoD)**: shared completeness checklist.
- **Empiricism**: transparency → inspect → adapt.

6) The Agile team & tools

- **Roles** (Scrum): **Product Owner** (value), **Scrum Master** (flow), **Developers** (build & test).
- **Kanban** teams have fewer prescribed roles—optimize **flow**.

- Tools: **Jira, Trello, Azure Boards, GitHub Projects, GitLab, Notion**, plus **CI/CD** and **feature flags**.

7) Agile Smart Pack (think: starter templates)

- **Templates** you'll reuse: Story template, Acceptance Criteria (Given/When/Then), DoD, DoR, Sprint Goals, Retro notes, Kanban board columns.
- Keep them **lightweight and visible** (Confluence/Jira).

8) Agile rituals & myths

- **Scrum events**: Sprint Planning, Daily Scrum, Review, Retrospective.
- **Kanban cadences**: Replenishment, Delivery review, Ops review, Retro.
- **Myths**: "No planning" (false—Agile plans a lot, just **smaller & continuous**), "No documentation," "Only for dev."

9) Great tools you can use

- Planning: **Jira, Azure Boards, Trello, ClickUp, Asana, Notion**.
- Collaboration: **Confluence, Miro/Mural**.
- Dev workflow: **GitHub/GitLab/Bitbucket**, **CI/CD** (GitHub Actions, Jenkins), **feature flags** (LaunchDarkly).
- Observability: **Grafana, Datadog**, Crashlytics (Android).

10) The best free tool (practically)

- For most teams: **Jira Free**, **Trello**, or **GitHub Projects** to start.
  Choose based on where your **code lives** and what your **org already uses** (Tesco → likely Jira + Confluence + Git platforms).

11) Real-world Agile Kanban board (IT)

- Columns often: **To Do → In Progress → Code Review → Test → Ready for Release → Done**.
- **WIP limits** to prevent overload (e.g., only 3 in "In Progress").
- Track **lead time** (idea → done) and **cycle time** (start → done).

12–14) Real-world non-IT Agile examples

- **Event planning** (wedding): backlog (venue, invites, catering), weekly check-ins, demo (menu tasting), retro (what to improve).
- **Marketing campaign**: hypothesis → run A/B ads → measure → iterate.
- **Home renovation**: deliver room-by-room (increments), adjust based on budget feedback.

15) Quick Jira tour

- **Project types**: Team-managed vs Company-managed.
- **Issue types**: Epic, Story, Bug, Task, Sub-task.
- Backlog, Sprint board/Kanban, **Reports** (Burndown, Velocity, CFD, Control Chart).
- **Workflows** & **Permissions** (who can move which status).

16) Using ChatGPT in Agile PM

- **Backlog help**: turn raw ideas into well-formed **user stories** with **acceptance criteria**.
- **Refinement**: split large stories, propose **MVP slices**.
- **Risk/assumption mapping**: list risks, test ideas for **spikes**.
- **Retro booster**: anonymized theme suggestions, **retro prompts**.
- **Comms**: draft concise **release notes**, stand-up updates.

17) Course project (practical activity idea)

- Build a small **MVP** (e.g., Android “Grocery List” app).
    1. Write 6–10 stories, 2) Map to a Kanban/Scrum board, 3) Define DoD, 4) Ship a login + list view in 1–2 weeks, 5) Run a retro.

18) Kanban in Microsoft Planner (or Trello)

- Buckets = columns, cards = tasks, labels = themes.
- Assignments, due dates, checklists, and **charts** to see flow.
- Start simple: **To Do / Doing / Review / Done** + **WIP limits**.

19) Fun & games

- Estimation games (Planning Poker), **“Buy a Feature”** prioritization, **Marshmallow challenge** (iteration & learning).

20) Before you start & how to start Agile

- **Pick a team cadence** (Scrum 1–2 weeks) or **Kanban**.
- Create an initial **MVP roadmap** (3–4 iterations).
- Set **working agreements** + DoD + Definition of Ready.
- Make work **visible** (board), start small, **inspect & adapt**.

21) Agile roles (simple)

- **Product Owner**: owns **value**, orders backlog, says **“not now.”**
- **Scrum Master / Flow coach**: removes impediments, improves **flow**, coaches on Agile.

- **Developers**: build, test, integrate, demo; collectively own **quality**.

22) Agile jobs & salaries (what matters to get hired)

- Roles: **Scrum Master**, **Product Owner**, **Agile Coach**, **Delivery Lead**, **Kanban Flow Manager**, **BA/QA** with Agile skills, **Engineers** with Agile delivery.
- **Signals** recruiters seek: shipped increments, real metrics (lead time, DORA), facilitation skills, Jira/Confluence competency, servant leadership.

23) Case study: breaking down a complex project

- Start with **outcome** → epics → thin vertical slices.
  Example (Android "Tesco Offers"):
  Epic: Personalized Offers → Slices: login → fetch offers API → display list → apply coupon → analytics event → accessibility polish.
- Each slice is **valuable on its own** and demo-able.

24) Agile recap

- Deliver value **early**, **often**, **safely**.
- Make work **visible**, keep it **small**, **finish before starting new**.
- Learn continuously: **experiments → feedback → adapt**.

25) FAQs article (typical answers you'll give)

- **How do we estimate?** Use **relative sizing** (story points) or **flow metrics** (cycle time) and forecast probabilistically.
- **When docs?** When they reduce risk or increase clarity—keep them **short, living**.
- **Sprints vs Kanban?** Sprints give **time-boxes & goals**; Kanban gives **continuous flow**—both are Agile.

26) Bonus lecture (evergreen advice)

- **Simplicity is a superpower**: ship the smallest thing that proves value.
- Invest in **quality** (testing, CI/CD) to go **faster long-term**.
- Protect **focus**: WIP limits, say **"not now"** to keep throughput high.

Handy mini-templates

User Story + AC

```
As a <type of user>,
I want <capability>,
so that <benefit>.

AC:
- Given <context> When <action> Then <observable result>
- ...
```

Definition of Done (example)

- Code reviewed, unit/UI tests passing, accessibility checks, feature flags, analytics events defined, merged to main, deployed to test, release notes updated.

Sprint Goal (example)

- “Enable biometric login for returning shoppers to reduce sign-in time by 50%.”

Kanban WIP policy (example)

- Max 3 items in “In Progress,” 2 in “Code Review,” 2 in “Test.” If a column is full, **swarm to unblock**.

Quick Jira mapping to your board stages

- **TO DO**: ordered stories with AC & DoR.
- **IN PROGRESS**: active development (pair/mob if blocked).
- **DESIGN REVIEW**: UX/Design approves UI flows & assets.
- **IN REVIEW**: code review (PR), static analysis, unit tests.
- **READY FOR TEST / IN TEST**: QA + exploratory; fix/loop.
- **READY FOR DEPLOY**: on main, packaged, release notes.
- **DONE**: deployed, monitored, analytics checked.

What to practice this week (fast wins)

1. Take one feature you’re building—**rewrite it as 3–5 thinner slices**.
2. Add **acceptance criteria** and a **DoD** to each slice.
3. Put them on a **board** with sensible **WIP limits**.
4. Do a **mini-retro** Friday: what to start/stop/continue?

User stories & backlogs

26) User Stories — what & why (3 min)

- **What:** Short, value-focused requirement written from a user's view.
  Format: As a <user>, I want <capability> so that <benefit>.
- **Why:** Aligns work to outcomes, invites conversation, and is easy to slice.

Example (Android)
As a shopper, I want biometric login so that I can sign in quickly.

27) Acceptance Criteria (AC) — make stories testable (2 min)

- Clear conditions that must be true for the story to be "done."
- Prefer **Given/When/Then**.

AC example

- **Given** biometrics are available **When** I tap "Use Face ID" **Then** I'm signed in and see Home.
- **Given** device has no biometrics **When** I tap "Use Face ID" **Then** I see a PIN fallback.

28) Writing Great User Stories (2 min)

- One user, one outcome; avoid "and/also."
- Keep INVEST: **I**ndependent, **N**egotiable, **V**aluable, **E**stimable, **S**mall, **T**estable.
- Add UX notes, edge cases, analytics, and non-functional needs (perf, a11y) as **notes**, not bloating the story text.

29) User Story Examples (2 min)

- **Search**: *As a shopper, I want to search products so that I can find items fast.*
- **Promo**: *As a Clubcard user, I want to see my active coupons so that I can save at checkout.*

30) Product Backlog vs Sprint Backlog (2 min)

- **Product Backlog:** PO-ordered list of **all** ideas (epics, stories, spikes). Changes often.
- **Sprint Backlog:** The team's **commitment for this sprint** + the plan (tasks). Changes only by the team.

31) Working Agreements (3 min)

- Team-made rules for collaboration. Keep visible.
  Examples:
- Stand-up at 10:00, cameras on.

- No merging without 1 review + all checks green.
- “Stop starting, start finishing” (respect WIP limits).

32) Definition of Ready (DoR)

- Entry checklist to pull a story into a sprint: user value clear, AC written, dependencies known, small enough, test approach known.

33) Definition of Done (DoD)

- Exit checklist to call work “done”: code reviewed, tests pass in CI, a11y checks, analytics event added, feature flagged, merged to main, release notes updated.

34) DoR vs DoD — casual compare

- **DoR** = “Ready to start?” **DoD** = “Truly finished?”
- Use both to reduce thrash and rework.

35) Product Increment (3 min)

- The integrated, tested product **at the end of a sprint**—potentially shippable (even if you choose not to release).

Estimation & forecasting

36) Introduction to Estimating (2 min)

- Estimate **relative size/complexity**, not exact hours. Forecast with **data** (velocity, cycle time).

37) Estimating in Agile (5 min)

- Two families:
  - **Scrum** → Story points (Fibonacci) + velocity.
  - **Kanban** → Cycle time & throughput.

38) Why Estimate? (3 min)

- Shared understanding, rough forecasting, risk surfacing, scope slicing—not for performance grading.

39–40) Techniques (Part 1 & 2) (4 + 5 min)

- **Planning Poker** (Fibonacci 1–2–3–5–8–13)
- **Affinity/Relative sizing** (group by similarity)
- **T-Shirts** (XS-S-M-L-XL) for epics
- **Triangulation** (compare to known reference stories)
- **#NoEstimates** (use cycle-time forecasts) when you have strong flow metrics

Tips: Time-box debates; if >13 pts, **split**.

People around the team

41–49) Other Roles (4–1 min each)

- **Project Sponsor:** funds/owns outcome; removes org-level blockers.
- **Business Leaders:** align product with strategy; protect focus.
- **Tech Leaders:** architecture, quality bar, developer experience.
- **Agile Detractors (Leadership):** often fear loss of predictability—win them with **transparent metrics** (lead time, DORA), small safe bets.
- **SME / Senior User:** deep domain input; availability matters.
- **Business Users:** source of feedback; involve them in reviews/betas.
- **Agile Coach:** improves flow & behaviors; systems thinker.
- **Apply to other frameworks:** concepts (stories/AC/DoD/metrics) carry to XP, Lean, SAFe, etc.

Flow & charts

50) Team Velocity (2 min)

- Average **completed** points/sprint. Use to forecast **your** team only (don't compare teams).

51) Burndown Chart

- Remaining work vs time. Healthy line "burns" down to zero.
  Smells: flat line (blocked), "cliff" at end (late testing), scope jumps (added work).

52) Burnup Chart (4 min)

- Completed work vs **total scope**—shows scope change clearly. Great with shifting backlogs.

53) Charts with non-Fibonacci (2 min)

- Linear scales (1–10) are fine if **consistent**. Relative sizing > exact numbers. Coarser scales reduce false precision.

54) Who updates charts? (1 min)

- **The team's tools** (Jira) update from issue status. Scrum Master ensures usage; team ensures **timely transitions**.

Kanban essentials

58) Intro & Four Principles (3 min)

1. Start with what you do now
2. Agree to pursue incremental change
3. Respect current roles
4. Encourage leadership at every level

59) Kanban Board (7 min)

- Visualize workflow: **To Do → In Progress → Review → Test → Done**.
- Use **WIP limits** to prevent overload and shorten cycle time.

60) Kanban Cards (1 min)

- Each card = one work item; include owner, AC, class of service, due date if needed.

61) Six General Practices (7 min)

1. Visualize work
2. Limit WIP
3. Manage flow
4. Make policies explicit
5. Implement feedback loops
6. Improve collaboratively, evolve experimentally

62) Key Metrics (3 min)

- **Cycle time**, **Lead time**, **Throughput**, **WIP**, **WIP age**, **Flow efficiency**.

63) Metrics by Board Type (2 min)

- **Dev boards:** cycle time & WIP age.
- **Support/Ops:** class of service, expedite lane, due-date performance.

64–67) Trello/Jira demos (6–4 min)

- Mirror your true workflow; don't over-complicate statuses.
- Use **automation** (move to "In Review" on PR open).

68) Kanban Board Statuses (2 min)

- Keep lean (5–7 columns). If queues form, create a policy/WIP limit for that step.

69) Swimlanes (2 min)

- Prioritize lanes: **Expedite**, **Bugs**, **Features**, **Blocked**; or by **Epic**/**Team**/**Service class**.

70) Steps to Implement Kanban (2 min)

1. Map your flow
2. Set initial WIP limits
3. Define policies (DoD per column)
4. Start measuring; adjust every retro

71–73) Conversation / Conclusion / Checklist (8 + 1 + 1 min)

- Start small; tune WIP; watch cycle time trend; celebrate flow improvements.

Scrumban (Scrum + Kanban)

74) Overview (4 min)

- Use Scrum's roles/reviews with Kanban's continuous flow & WIP limits.

75) Sprint debate (1 min)

- Keep sprints if goals help; drop them if **flow** and **replenishment cadences** suffice.

76) Continuous Flow (2 min)

- Pull next ready item immediately; demo often; plan by **replenishment meeting**.

77) Timebox (4 min)

- Timebox **reviews/retros** even if delivery is continuous.

78) Feature Freeze (1 min)

- Pause new work before release; stabilize; then resume flow.

79) Target Audience (3 min)

- Great for **maintenance/ops**, **teams with frequent interrupts**, or **Scrum teams** wanting smoother flow.

80–84) Puzzles, thanks, what's next (short)

- Practice terms, then keep learning.

"What's next?" — concrete next steps for you

In one sprint (2 weeks):

1. **Rewrite 8–10 stories** with AC using Gherkin; add DoR/DoD checklists in Jira.
2. **Run Planning Poker**; adopt Fibonacci; split anything >8 pts.
3. **Set WIP limits** (e.g., 3 in "In Progress," 2 in "Review").
4. Turn on **Jira Burnup** for the sprint and review it in stand-ups.
5. Do a **retro**; pick **2 small actions** (owner, due date).

If you want, I can convert this into:

- a **Confluence page** with copy-paste story/AC/DoR/DoD templates, or

- a **Jira workflow/board** suggestion mapping (TO DO → IN PROGRESS → DESIGN REVIEW → IN REVIEW → IN TEST → READY FOR DEPLOY → DONE) with WIP limits and automation examples.

Agile Fundamentals Deep Dive (Scrum • Kanban • Scrumban)

Tailored for day-to-day engineering with Android/Kotlin + Jira examples. Use this as a reference, onboarding handout, or interview prep.

Contents

A) User Stories & Backlogs (26–35)

26) User Stories — What & Why

Definition: A concise statement of user value, written from the user's perspective.

Format:
As a <type of user>, I want <capability> so that <benefit>.

Why it works:

- Centers **outcomes** (value) over outputs (tasks).
- Encourages conversation and discovery.
- Easy to slice into smaller increments.

Android/Jira Example:
As a returning shopper, I want to sign in with biometrics so that I can access my list quickly.

AC and tasks are captured in the issue; UX mock is linked; analytics event named login_success_biometric.

27) Acceptance Criteria (AC) — Make It Testable

Definition: Conditions that must hold true for the story to be accepted.

Structure (Gherkin):

- **Given** context
- **When** action
- **Then** observable result

Example AC:

- **Given** device supports biometrics **When** I tap "Use Face ID" **Then** I see Home within 2s.
- **Given** biometrics fail 3 times **When** I try again **Then** I am offered PIN fallback.
- Analytics event login_biometric_clicked fires once per attempt.

28) Writing Great User Stories

INVEST qualities: Independent, Negotiable, Valuable, Estimable, Small, Testable.

Tips:

- One user, one outcome; avoid "and/also".
- Keep story text short; put details in **AC/notes**.
- Capture non-functionals (perf, a11y, security) in AC or DoD.
- Add **edge cases** and **analytics** explicitly.

Anti-patterns: solution-first ("Create 4 tables in DB"), role soup ("As a system…"), vague benefits.

29) User Story Examples (Good vs Weak)

- **Good:** *As a shopper, I want to scan a barcode so that I can add items faster.*
    - AC: supports offline, invalid barcode error, adds within 500ms after scan.
- **Weak:** *Add barcode library to app.* (task; not user value)

30) Product Backlog vs Sprint Backlog

- **Product Backlog (PB):** Ordered list of all work ideas (epics/stories/spikes). Owned by **PO**. Changes frequently.
- **Sprint Backlog (SB):** PB items **selected** for the sprint + plan to deliver them. Owned by **team**.

Jira mapping: PB = backlog view; SB = active sprint board.

31) Working Agreements

Team-created collaboration rules, visible and short. Review in each retro.

Examples:

- Daily at 10:00, 15 min, blockers first.
- PR needs 1 reviewer + CI green; pair on critical fixes.
- "Stop starting, start finishing" → respect WIP limits.
- Quiet focus hours 2–4pm; Slack mentions only for blockers.

32) Definition of Ready (DoR)

Entry checklist to start work. Aims to reduce churn.

Sample DoR:

- User, outcome, and **AC** are clear; mocks linked.
- No major unknown dependency; data contracts known.
- Small enough (< 2–3 days of dev time or < 8 pts).
- Test approach noted (unit/UI), analytics defined.

33) Definition of Done (DoD)

Exit checklist to call work "done". Prevents hidden work.

Sample DoD (mobile):

- Code reviewed; static analysis clean.
- Unit tests + UI smoke pass in CI; coverage unchanged or higher.
- Feature flag off by default; analytics events verified.
- A11y labels, dynamic type supported; perf budget met.
- Merged to `main`; release notes + owner docs updated.

34) DoR vs DoD (Compare)

- **DoR:** "Ready to start?"
- **DoD:** "Truly finished?"
  Use both. If items bounce in QA, strengthen DoR/DoD.

35) Product Increment

Definition: The integrated, tested product at end of the sprint—**potentially shippable**. You may hold release with a flag, but the increment must be releasable in principle.

Practice: Demo from `main` build, not a branch.

B) Estimation & Forecasting (36–40) + Charts (50–55)

36) Intro to Estimating

Estimate **relative size/complexity**, not hours. Use history to forecast. Optimize **flow**, not "accuracy".

37) Estimating in Agile

Scrum path: Story points (often Fibonacci: 1,2,3,5,8,13). Forecast via **velocity** (avg points done per sprint).
Kanban path: Don't estimate each story; forecast with **cycle time** and **throughput** using past data.

38) Why Estimate?

- Align understanding; surface risk/unknowns.
- Enable **thin slicing** and release planning.
- Provide **credible ranges** (not a single date!) to stakeholders.

39) Estimation Techniques (Part 1)

Planning Poker:

1. Discuss story; everyone picks a card privately.
2. Reveal together; outliers explain.
3. Re-vote; settle fast or spike.

Affinity Sizing:

- Sort stories by relative size quickly (S…XL) on a wall; assign points after grouping.

40) Estimation Techniques (Part 2)

T-Shirts: Use for epics/early discovery; convert later to points.
Triangulation: Compare to reference stories ("like the 3-point login story").
NoEstimates: Use **cycle-time percentiles** (e.g., 50% of items finish within 3 days; 85% within 6 days) for forecasting.

Rule of thumb: If >8 points or >3 days, **split** before pulling into a sprint.

50) Team Velocity

Definition: Completed story points per sprint (done to DoD).
Use: Internal planning only; don't compare across teams.
Stabilization: Usually after 3–5 sprints with stable team + DoD.

51) Burndown Chart

What: Remaining work vs time within a sprint.
Smells & Fixes:

- Flat line → blockers/WIP too high → swarm, cut scope.
- Late cliff → testing bunched → shift left, slice smaller.
- Upward jumps → scope added → re-negotiate or add capacity.

52) Burnup Chart

What: Work completed vs **total scope**.
Why better: Makes scope changes explicit; easier for stakeholders to see progress despite added work.

53) Using Charts with Non-Fibonacci

Linear scales (1–10) work if consistent. Communicate that the numbers are **relative**, not hours.

54) Who Updates Charts?

Tools (Jira) update charts as issues move columns. Team must **transition issues promptly**. SM ensures the hygiene; not a secretary.

55) ACTIVITY (Scrum Terms Worksheet)

Match terms to definitions; see **Section F** for a printable set + answers.

C) Roles Around the Team (41–49)

41) Overview of Other Roles

Beyond Scrum's PO/SM/Developers, several org roles interact with an Agile team. Clarity reduces misaligned expectations.

42) Project Sponsor

Funds and champions the outcome; removes organizational blockers; signs off on value metrics.

Engagement cadence: Monthly review on outcome metrics and key risks.

43) Business Leaders

Provide strategy and market context; make **trade-off decisions**; protect focus (say “not now” to distractions).

44) Technology Leaders

Own architecture guardrails, security, platform health; enable teams with CI/CD, test infra, and observability.

45) Agile Detractors — Leadership

Common concerns: predictability, compliance, “lack of control”.
Counter: Transparent **flow metrics** (lead time, WIP, DORA), frequent demos, risk-based plans, lightweight governance.

46) Subject Matter Expert / Senior User

Domain depth; ensure regulatory/industry nuances aren’t missed. Book **office hours** to avoid ad-hoc thrash.

47) Business Users

Provide feedback via usability sessions, betas, or staged rollouts. Close the loop with **analytics** and NPS.

48) Agile Coach

Improves behaviors and **flow**. Facilitates retros, sets up experiments, trains on practices, mentors SMs/POs.

49) Applying to Other Frameworks

- **XP:** Pairing, TDD, continuous integration → strengthen DoD.
- **Lean:** Waste elimination, small batch sizes → Kanban WIP limits.
- **SAFe/LeSS:** Multi-team coordination; keep local agility (thin backlogs, small batches) within scaling constraints.

D) Kanban Essentials (58–73)

58) Intro & Four Principles

1. Start with what you do now.
2. Agree to incremental, evolutionary change.
3. Respect current roles/responsibilities.
4. Encourage leadership at every level.

59) Kanban Board

Purpose: Visualize work and optimize **flow**.
Typical columns: To Do → In Progress → Review → Test → Ready for Release → Done.
Policies: WIP limits per column; entry/exit criteria per step.

60) Kanban Cards

Each card = one work item; include title, owner(s), AC link, class of service, due date if applicable.

61) Six General Practices

1. Visualize.
2. Limit WIP.
3. Manage flow (watch cycle time, WIP age).
4. Make policies explicit.
5. Implement feedback loops (daily flow review, weekly ops).
6. Improve collaboratively, evolve experimentally.

62) Kanban Key Metrics

- **Lead Time:** request → done.
- **Cycle Time:** start → done.
- **Throughput:** items done per time unit.
- **WIP:** items in progress.
- **WIP Age:** time since start; high age = risk.
- **Flow Efficiency:** (active time / total time).

Targets: Lower cycle time, stable throughput, balanced WIP.

63) Metrics Based on Board Type

- **Dev/Product:** cycle time, WIP age, throughput, blocked time.
- **Support/Ops:** due date performance, expedite policy, resolution time bands.

64) Kanban Demo: Trello (Conceptual Mapping)

- Lists = columns; cards = items; labels for classes of service (e.g., Bug, Feature, Expedite). Use checklists and due dates.

65) ACTIVITY: Setup Trello

- Create lists: Backlog / Ready / Doing (WIP 3) / Review (WIP 2) / Done.
- Add 10 sample stories; set labels; assign owners; enable calendar view.

66) Follow-along Demo #2 (Jira Quickstart)

- Project → Kanban template.
- Board settings: add WIP limits; map statuses; enable Control Chart & CFD.
- Automation: move to **In Review** when PR opened; to **Done** when merged.

67) Kanban Demo: Jira (Tips)

- Keep statuses lean (5–7).
- Use **Quick Filters**: assignee = currentUser(), type in (Bug), priority = Highest.

68) Kanban Board Statuses

Name by **state**, not departments. Avoid "Design → Dev → QA" silos; prefer flow stages like **Discover / Build / Validate**.

69) Swimlanes

Separate by class of service (Expedite, Fixed Date, Standard, Intangible) or by epic/theme. Keep lanes few (≤4).

70) Steps to Implement Kanban

1. Map current workflow with the team.
2. Set initial WIP limits (start tight; adjust).
3. Define entry/exit policies per column.
4. Start measuring; review weekly; run targeted experiments.

71) Casual Conversation about Kanban (FAQ)

- **Q:** Can we estimate? **A:** Optional; rely on flow metrics first.
- **Q:** How to handle interrupts? **A:** Expedite lane with strict policy & capacity guardrail.
- **Q:** Do we need stand-ups? **A:** Yes—make them **flow reviews**.

72) Concluding Kanban

Kanban optimizes **time-to-value** via small batches and visual control. Focus on **finishing**, not starting.

73) Getting Started Checklist

- Map flow; create board.
- Set WIP limits; publish policies.
- Define Done per column.

- Turn on Control Chart & CFD.
- Review WIP age daily; swarm on aging items.
- Run weekly ops review; monthly retro on metrics.

E) Scrumban (74–80)

74) Overview

Blend Scrum’s roles & cadences with Kanban’s continuous flow. Keep sprints if goals help; otherwise, use **replenishment**.

75) Scrumban: Sprint (Debate)

- Keep sprints if stakeholders value a **sprint goal** and demo cadence.
- Drop sprints if work is highly interrupt-driven; use **service-level expectations (SLEs)** instead.

76) Continuous Flow

Pull the next **ready** item immediately. Demo often (e.g., biweekly). Use replenishment meeting to refill **Ready** column.

77) Timebox

Timebox planning/reviews/retros even with continuous delivery (e.g., 60-min review every 2 weeks).

78) Feature Freeze

Prior to a major release, pause new work; focus on stabilization, perf, a11y, rollout comms, and monitoring.

79) Target Audience

Great for ops/maintenance teams, platforms, or product teams wanting smoother flow without losing stakeholder cadence.

80) Casual Conversation (Common Qs)

- **Q:** Can we have a sprint goal and WIP limits? **A:** Yes—sprint goal for focus; WIP limits for flow.
- **Q:** How to forecast? **A:** Use throughput + cycle-time distributions.

F) Activities & Quizzes (with printable sets)

55/56/57) Scrum Terms Worksheet + Solutions

Match the term to the definition (answers below):

1. Product Increment
2. Definition of Done
3. Definition of Ready
4. Sprint Backlog
5. Velocity
6. Acceptance Criteria
7. Burnup Chart
8. Burndown Chart
9. Working Agreement
10. Spike

Definitions:
A) Team rules for collaboration
B) Conditions to accept a story
C) Remaining work vs time in a sprint
D) Investigative time-boxed research
E) PB items selected + delivery plan
F) Integrated, potentially shippable product
G) Checklist to start a story
H) Avg points completed per sprint
I) Completed work vs total scope
J) Exit checklist for "done"

Answer Key: 1-F, 2-J, 3-G, 4-E, 5-H, 6-B, 7-I, 8-C, 9-A, 10-D

65/66) Trello/Jira Hands-on

- Build the board specified in **D-65** and **D-66**.
- Populate with 10 user stories (biometrics login slice pack).
- Measure first week's cycle time; set an initial **SLE** (e.g., 85% of items ≤ 5 days).

73) Kanban Starter Checklist

Printable list provided in **D-73**; copy to your team's Confluence.

81/82) Agile Terms Crossword (Lite Alternative)

Word list to self-test: Agile, Scrum, Kanban, Scrumban, Backlog, Sprint, Retro, DoD, DoR, Velocity, Burndown, Burnup, WIP, CycleTime, Throughput, Epic, Story, Spike, AC, CFD. Challenge: Write a one-line definition and a team example for each.

G) What’s Next & Resources (Bonus / Next Steps)

Continuing Education

- **Practice > theory:** Run one small improvement experiment every sprint (e.g., lower WIP in Review from 3 → 2).
- **Certifications:** ICAgile ICP (foundations) or **ICP-APO** (Product Ownership) if product-focused.
- **Communities:** Join an Agile meetup or internal CoP; present a 5-minute case study on your team’s flow metrics.

Personal Roadmap (4 weeks)

- **Week 1:** Rewrite top 10 stories with AC; set DoR/DoD; enable Jira burnup.
- **Week 2:** Planning Poker; split >8pt items; WIP limits.
- **Week 3:** Introduce Control Chart review in stand-ups; track WIP age.
- **Week 4:** Retro on metrics; choose two process tweaks.

Templates (copy/paste)

User Story
As a <user>, I want <capability> so that <benefit>.

Acceptance Criteria (GWT)

- Given <context> When <action> Then <result>

DoR (checklist)

- Value clear • AC present • Size small • Dependencies known • Test plan & analytics noted

DoD (checklist)

- Review ✓ • Tests/CI ✓ • Flag ✓ • A11y/perf ✓ • Docs/notes ✓ • Merged to main ✓

Kanban Policies

- WIP: In Progress ≤ 3; In Review ≤ 2.
- Expedite lane ≤ 1; must include customer impact and rollback.

Android/Jira Mini-Example Pack (Biometric Login)

Stories: capability detection, enable prompt, error & fallback, analytics event, accessibility audit, experiment rollout.
AC highlights: happy path ≤ 2s, fallback offered, event names fixed, TalkBack labels present.
DoD: unit/UI tests pass, Crashlytics monitored, feature flag default off.
Board: To Do → In Progress (WIP 3) → In Review (WIP 2) → In Test (WIP 2) → Ready for Release → Done.
Metrics: Cycle time p50 ≤ 3d; WIP age alert at 4d.

End of guide.